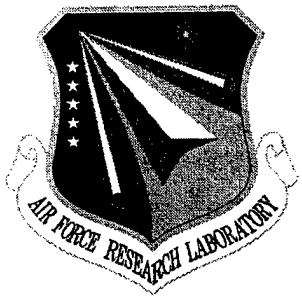
AFRL-IF-RS-TR-1999-31
Final Technical Report
February 1999



EMBEDDED AND REAL-TIME APPLICATION OF HIGH-PERFORMANCE SCALABLE COMPUTING

Lockheed Martin

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. D354

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1999-31 has been reviewed and is approved for publication.

APPROVED:

RALPH KOHLER
Project Engineer

FOR THE DIRECTOR:

NORTHRUP FOWLER, III, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTC, 26 Electronic Parkway, Rome, NY 13441-4514. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

EMBEDDED AND REAL-TIME APPLICATION OF HIGH-PERFORMANCE SCALABLE COMPUTING

Ronald E. Hamlet

Contractor: Lockkheed Martin
Contract Number: F30602-95-2-0039
Effective Date of Contract: 14 September 1995
Contract Expiration Date: 5 September 1998
Short Title of Work: Embedded and Real-Time Application of High Performance Scalable Computing

Period of Work Covered: Sep 95 - Sep 98

Principal Investigator: Ronald E. Hamlet
Phone: (315) 456-0123
AFRL Project Engineer: Ralph Kohler
Phone: (315) 330-2016

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Ralph Kohler, AFRL/IFTC, 26 Electronics Parkway, Rome, NY 13441-4514.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1999	Final Sep 95 - Sep 98

4. TITLE AND SUBTITLE
EMBEDDED AND REAL-TIME APPLICATION OF HIGH-PRFORMANCE SCALABLE COMPUTING

5. FUNDING NUMBERS
C - F30602-95-2-0039
PE - 62301E
PR - D002
TA - 01
WU - P4

6. AUTHOR(S)
Ronald E. Hamlet

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin
Ocean, Radar and Sensor Systems
P.O. Box 4840
Electronics Parkway
Syracuse NY 13221-4840

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Air Force Research Laboratory/IFTC
26 Electronics Parkway
Rome NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-1999-31

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Ralph Kohler/IFTC/(315) 330-2016

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Pre and Post Doppler Space-Time Adaptive Processing (STAP) architectures were considered for target implementations based on embedded High Performance Scalable Computing (HPSC) architectures leveraging commercially available processing technology from Analog Devices Super Harvard Architecture (SHARC) Digital Signal Processor (DSP). Algorithm partitioning and mapping was performed that demonstrated initial feasibility and then a sizing study was performed for a theoretical implementation. Further modeling and simulation studies utilized a discrete event simulator to perform detailed timing analysis and three different mappings of the Recursive Modified Gram Schmidt with Error Feedback (RMGSEF) algorithm in order to obtain insight into processor communication utilization and data latency. This effort culminated in a real time Radar demonstration of the RMGSEF algorithm that was implemented using parallel SHARC processors based on the High Performance Scalable Computer (HPSC) to perform the QR Decomposition (QRD). The demonstration Radar System incorporated 18 antenna elements over three pulse repetition intervals resulting in 54 degrees of freedom with performance of less than 15 ms of latency and a 42KHz sample rate. Further studies concentrated on alternative STAP solutions based on evolving Motorola PowerPC's and Field Programmable Gat Arrays (FPGAs).

14. SUBJECT TERMS
Space-Time Adaptive Radar, Airborne Early Warning Radar

15. NUMBER OF PAGES
60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Space-Time Adaptive Processing using a high performance scaleable computer was demonstrated under a cooperative agreement with DARPA to establish the application of embedded HPSC technology to a real time radar problem. This effort advanced the HPSC technology through hardware and software development support.

The program consisted of three parts followed by a no cost extension.

The first consisted of an Architecture Study to determine metrics associated with implementation of two STAP radar-processing architectures.

The second investigated the tools associated with application development. Given the immaturity of the HPSC technology at the time of the study, this effort also applied RASSP tools to predict processing performance in terms of efficiency and latency.

The third part consisted of a hardware/software demonstration where by the Adaptive Array Processor was selected for implementation. The Recursive Modified Gram Schmidt algorithm was presented as the required algorithm with very strict update and low latency requirements. This processor application was developed and successfully demonstrated a real time radar confirming that all requirements were met and that the predicted performance from the modeling effort was realized.

The HPSC demonstration was followed by a no cost extension that was comprised of three studies.

The first study of the extension was an architecture study to evaluate emerging commercial processing technologies leveraging HPSC and their application to STAP. For the initial demonstration, the HPSC implemented Adaptive Array Processor was required to interface to an existing Digital Beamformer which dictated the latency of the adaptive solution. An implementation based on HPSC was investigated and compared to Field Programmable Gate Array (FPGA) based solutions.

The second study of the extension was an investigation of a Switching Network to provide sensor data to a Myrinet™ network of processors. Here DARPA's Reconfigurable Transport Engine (RCTE) technology is being exploited to provide a low latency interface to a high performance network.

The third study of the extension was to determine the latency of adaptive processing solutions based on Motorola's emerging PowerPC technology for real-time signal processing. Here, different communication approaches were explored and compared.

2 Processing Architectures

A Pre-Doppler and Post-Doppler processing architecture was evaluated for implementation consideration. The Pre-Doppler processing architecture performs a reduced dimension space-time adaptive nulling function through adaptive combination of the input antenna element data. This processing architecture is classified as Pre-Doppler STAP since the Adaptive Beamforming is performed on the data prior to coherent integration, or Doppler Filtering. The high level functionality of this benchmark includes Adaptive Weight Computation, Beamforming, Pulse Compression, Doppler Processing, and CFAR Processing.

The input data set designed for use with this benchmark consists of a three-dimensional data-cube, of 18 antenna elements x 18 PRI's x 36000 range gates, to be processed within a target 73 ms Coherent Pulse Interval (CPI) time.

The processing bandwidth that is required to implement this processing architecture is approximately 25 Gflops.

Figure 1 illustrates the input data cube as well as the processing function chain for this architecture

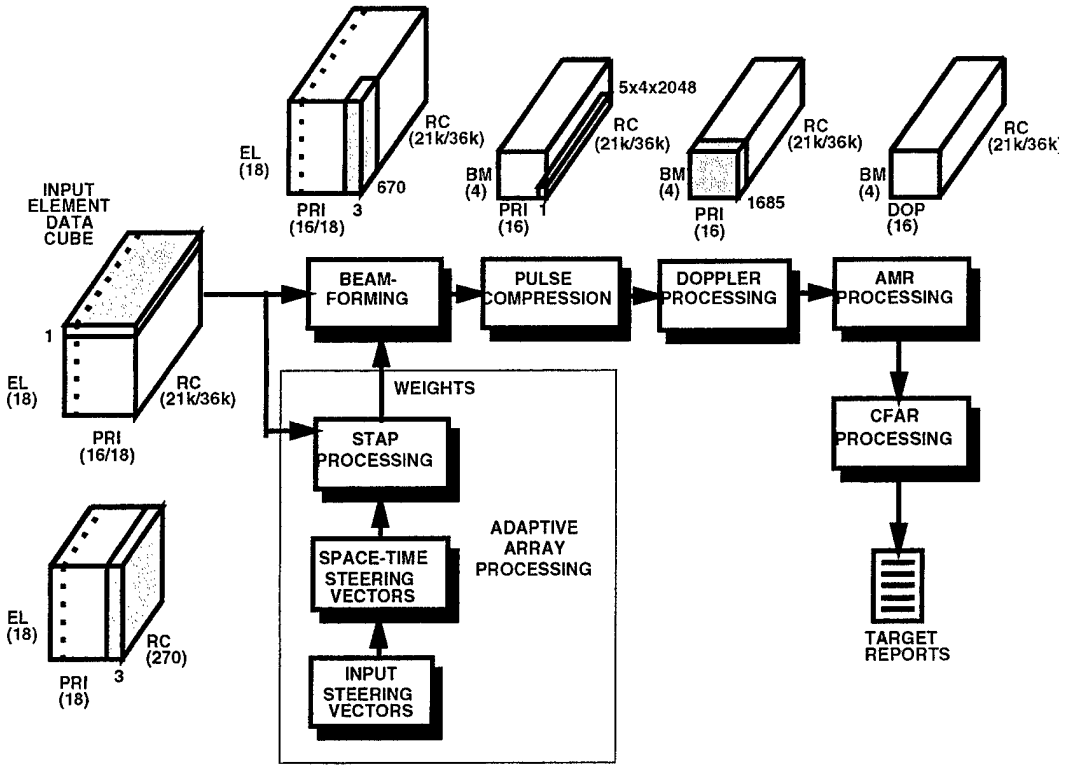


Figure 1- Pre-Doppler STAP Architecture

The second processing architecture considered performs a reduced dimension space-time adaptive nulling function through adaptive combination of the input beam space data and is classified as Post-Doppler STAP since Doppler Processing is the first function performed.

The high level functionality of this benchmark includes Doppler Processing, Adaptive Weight Computation and Beamforming, Pulse Compression, and CFAR Processing.

The input data set designed for use with this benchmark consists of a three-dimensional data-cube, 18 elements x 18 PRIs x 36000 range gates, to be processed within the target 73 ms CPI time.

The processing bandwidth that is required to implement this processing architecture is approximately 32 Gflops.

Figure 2 illustrates the input data cube as well as the processing function chain for this architecture.

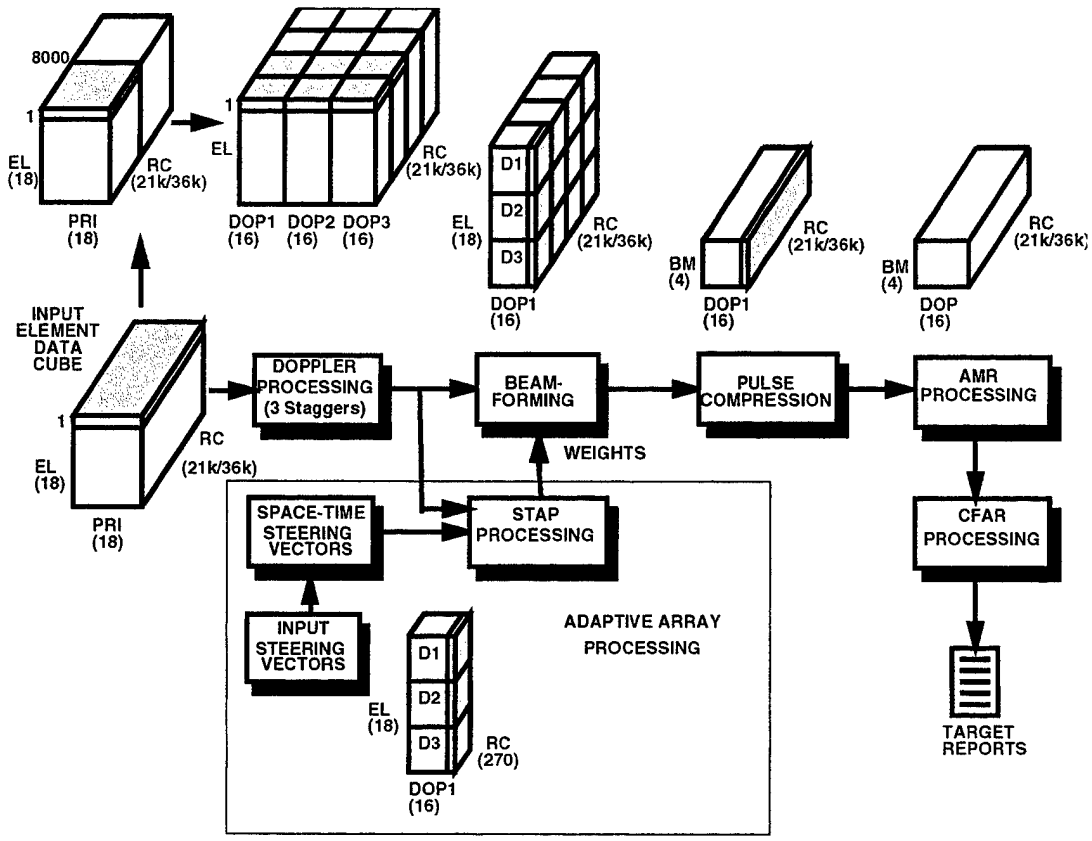


Figure 2 - Post-Doppler STAP Architecture

3 Architecture Definition Methodology

The hardware Architecture Definition Methodology is performed in two stages. The first stage is the Mapping and Partitioning Feasibility Study that maps selected Radar Processing architectures onto the processing platform. The second stage is the Detailed Modeling and Timeline Simulation that takes selected processing functions and performs a detailed analysis in order to refine estimates created in the first stage mapping.

The mapping and partitioning task determines the number of processors required for performing the specified processing and also accounts for specified overhead requirements. As a result of this partitioning analysis, the following Technical Performance Measures (TPM's) are derived for each processing stage:

- Number of Nodes/Processors required.
- Number of simultaneous I/O connections required.
- Processing Throughput (floating point operations per second: flops)
- Efficiency (%)
- Latency (seconds)

The Detailed Modeling and Timeline Simulation task takes a portion of the overall processing architecture, namely the Recursive Modified Gram-Schmidt with Error Feedback (RMGSEF) QR Decomposition [3] function from the STAP processing, and performs a detailed timeline analysis to examine processor utilization and latency. This function was chosen because of the processing complexity and the desired requirement for a low latency solution.

The real-time radar demonstration targeted the RMGSEF algorithm with stringent latency (< 15 milliseconds) and sample rate (42 KHz) performance requirements. This effort included:

- Detailed design of the overall architecture including inter-processor communication and interfaces.
- Application algorithm mapping.
- Detailed software design and code.
- System implementation, integration and test.
- Application demonstration of topology and architecture.
- AEWSC analysis including processor benchmarks.

3.1 Mapping and Partitioning

The mapping and partitioning task determines the number of processors required for performing the specified processing and also accounts for specified overhead requirements. The decision to utilize either COTS or Custom components is also established at this time. Factors such as size, weight, data I/O and processing load are factors that are used as a guide to selecting either a COTS or Custom implementation. As a result of this partitioning analysis, the following Technical Performance Measures (TPM's) are derived for each processing stage:

- Number of Nodes/Processors required.
- Number of simultaneous I/O connections required.
- Processing Throughput (floating point operations per second: flops)
- Efficiency (%)
- Latency (seconds)
- Size, Weight, Power

3.2 Modeling and Simulation

The Detailed Modeling and Timeline Simulation task takes a portion of the overall processing architecture, namely the Recursive Modified Gram-Schmidt (RMGSEF) QR Decomposition function from the Adaptive Weight Processing, and performs a detailed timeline analysis to examine processor utilization and latency. This function was chosen because of the processing complexity and the requirement for a low latency solution.

The real-time radar demonstration targeted the RMGSEF algorithm with stringent latency (< 15 milliseconds) and sample rate (42 KHz) performance requirements. This effort included:

- Detailed design of the overall architecture including inter-processor communication and interfaces
- Application algorithm mapping
- Detailed software design and code
- System implementation, integration and test
- Application demonstration of topology and architecture
- AEWSC analysis including processor benchmarks

4 Algorithm Partitioning

Algorithm partitioning is the process whereby the target algorithms are examined for parallelism and the data is partitioned for data transfer balancing and node processor loading. The parallelism of an algorithm allows it to be computed by multiple processing nodes all operating in parallel on a specific portion of the total algorithm. The partitioning of the data is utilized in conjunction with the parallelism of the algorithm to determine the processing load for a node computing a specific portion of the algorithm. In addition, the data partitioning is used to balance the data throughput into and out of a processing node.

Estimated processing times are considered in relation to the processing basis time for that function, e.g., a PRI or CPI, and an equivalent number of processors is computed by dividing the processing basis by the estimate after adjusting for setup times

required for data input and other preprocessing. From these estimates, the processing efficiency and throughput may be estimated for each function using the basis times. From this a course number of processors are determined that will execute the algorithm in the time required.

The next step is to examine the data to identify a processor network that fully utilizes the available computational bandwidth. This is accomplished by analyzing the data flow though the algorithm and determining the computational and data transfer load associated with each function in the algorithm. In this manner "packets" of data, which are sub-cubes of the entire data cube can be mapped to a specific processing node so that a balance between the computational bandwidth and the data transfer bandwidth can be achieved. Figure 1 and 2 depict packet shapes for the various processing stages, indicated by the shading on the data-cubes for each function.

The processing timing for the SHARC processors was based on the *ADSP2106x Sharc User's Manual* [4] and the Ixthos *IXLibs-21k DSP Libraries User's Manual* [5]. These manuals provide timing estimates for various operations in numbers of instruction cycles and are converted to processing times. An overhead factor of 30% is used to account for processing code not included in the initial sizing estimates. It is assumed that the SHARC processors are running at 40 Mhz, which corresponds to a 25 ns instruction execution time.

The processing timing for the PPC processors was based on the PPC603e User's Manual and the Mercury DSP Libraries User's Manual. It is assumed that the SHARC processors are running at 200Mhz.

The Myrinet™ data channels are assumed to be capable of data throughput rates of 160 Mbytes/sec but for purposes of this analysis, this number is reduced to 120 Mbytes/sec to provide an engineering margin.

The RACEway™ data channels are assumed to be capable of data throughput rates of 160 Mbytes/sec but for purposes of this analysis, this number is reduced to 120 Mbytes/sec to provide an engineering margin.

The SHARC Link Port interfaces are assumed to be capable of data throughput rates of 40 Mbytes/sec, however, for purposes of this analysis, this number is reduced to 30 Mbytes/sec to provide an engineering margin.

5 Adaptive Array Processing Platform Architecture

Space-time Adaptive Processing (STAP) algorithm architectures for Airborne Early Warning (AEW) Radar applications are evaluated for implementation based on three unique hardware platforms. These are the High Performance Scalable Computer (HPSC) [1], Mercury Computer Systems SHARC VME module utilizing RACEway, and the Ixthos SHARC MCM VME module.

All of the three platforms enable the design of a programmable, scalable, expandable hardware architecture using a multi-processor environment. The primary difference among the platforms is in the quantity/type of DSP per module and the available interface options for the movement of the data. Each platform is implemented using either Analog Devices ADSP2106x Super Harvard Architecture (SHARC) processors or Motorola PowerPC processors connected in a multi-processor environment.

The HPSC architecture is implemented using 8 Analog Devices ADSP2106x Super Harvard Architecture (SHARC) processors connected in a multi-processor environment utilizing Myrinet™ switching technology. The Myrinet™ components provide a high-speed packet switching and routing capability using intelligent co-processors to perform data movement. This combination of SHARC processors for data processing and Myrinet™ LANai processors for data movement allows application code to be written which is scalable to different data-cube sizes and processing throughputs and allows the communications code to be de-coupled from the applications code. A processing node in the HPSC context is defined as a group of four SHARC processors, interconnected via a local bus to a common processing node storage memory and a LANai Myrinet™ interface chip. The LANai interface chips are then interconnected to a multi-port switch that provides interconnectivity to other processing nodes.

The Mercury Computer Systems architecture is implemented utilizing a VME Motherboard that can accommodate multiple compute nodes (CNs). Mercury offers i860, PowerPC, and SHARC daughter cards, each available in multiple configurations. These CN's enable various processing components to be added to the motherboard resulting in a configuration that meets the need of the system. Concurrent RACEway connections allow input, inter-processor communication, and output to occur simultaneously. The PowerPC Compute nodes were utilized on this platform. This provided a total of 4 PPC processors per module.

The Ixthos SHARC Muti-Chip Module (MCM) VME module contains 16 SHARC DSP's that are packaged utilizing MCM technology. Link ports are the interfaces that provide data transfer capability. In addition, the module provides two PMC locations for specialized I/O and or capability.

6 Adaptive Array Processing

The following section describes the architecture studies that were performed for the Adaptive Array processing. The architectures described here are based on the modeling and evaluation of the Recursive Modified Gram Schmidt algorithm for the generation of the Adaptive Weights.

6.1 Adaptive Array Processing on the HPSC Platform

The HPSC platform is based on the SHARC DSP as the compute engine and the Myrinet™ network as the data transport mechanism. The Myrinet™ network is a packet based communication network; data is sent as packets with header information including the packet destination and routing tables in the LANai processors determine the paths the packets take. The data partitioning into packets as described above further determines the network topology between processing stages, including the size in number of ports of the Myrinet™ switches and the number of connection paths to each processing node.

Interconnect diagrams detailing required numbers of processing nodes and Myrinet™ communication channels are created for these Pre and Post Doppler architectures. These interconnect diagrams are then mapped onto the HPSC hardware architectures in order to determine the following hardware metrics:

- Number of processor/communication boards
- Size
- Weight
- Power; Mflops/Watt

Figure 3 details the construction of an HPSC 3.3-volt multi-chip module (MCM), including eight SHARC processors, local memory and Myrinet™ interface units using Myricom LANai chips.

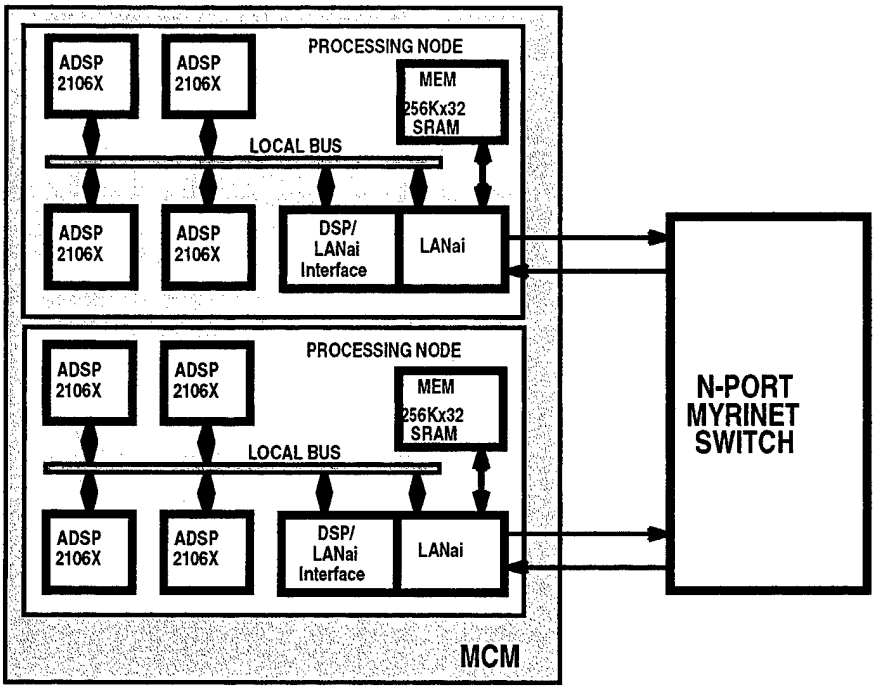


Figure 3 - HPSC 3V MCM Block Diagram

These modules are a target for a future development of the HPSC program and will be used to construct a 32-processor arithmetic processor (APU) board. These boards will be used in conjunction with Myrinet™ switching boards (XBAR), consisting of Myrinet™ crossbar switches and I/O buffers as illustrated in Figure 4, to construct an HPSC chassis.

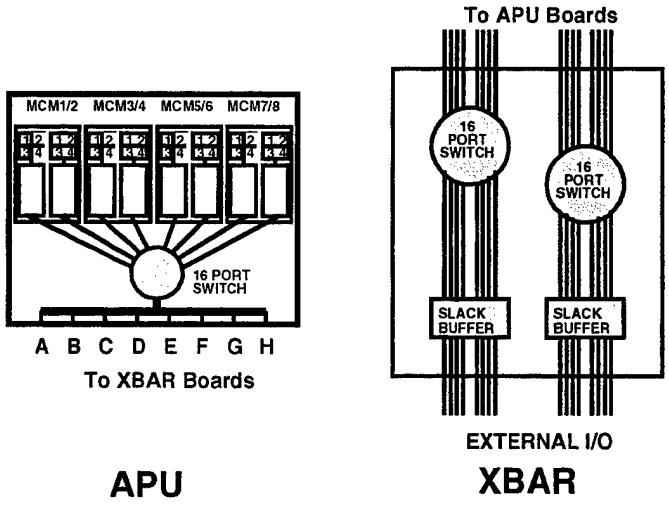


Figure 4 - HPSC Processor and Switching Boards

A combination Pre/Post Doppler HPSC architecture is next examined that can handle either configuration that utilizes a single hardware solution. The dual functionality is accomplished via software programmability of packet routing tables loaded into the LANai chips in each processing node which allow the data to be re-routed to implement one architecture or the other. This programmable functionality is an extremely desirable feature for next-generation radar systems since it demonstrates the flexibility in algorithm and architecture implementations possible in the entire radar processing chain following the A/D converters. Figure 5 illustrates the resulting combination Pre/Post Doppler architecture.

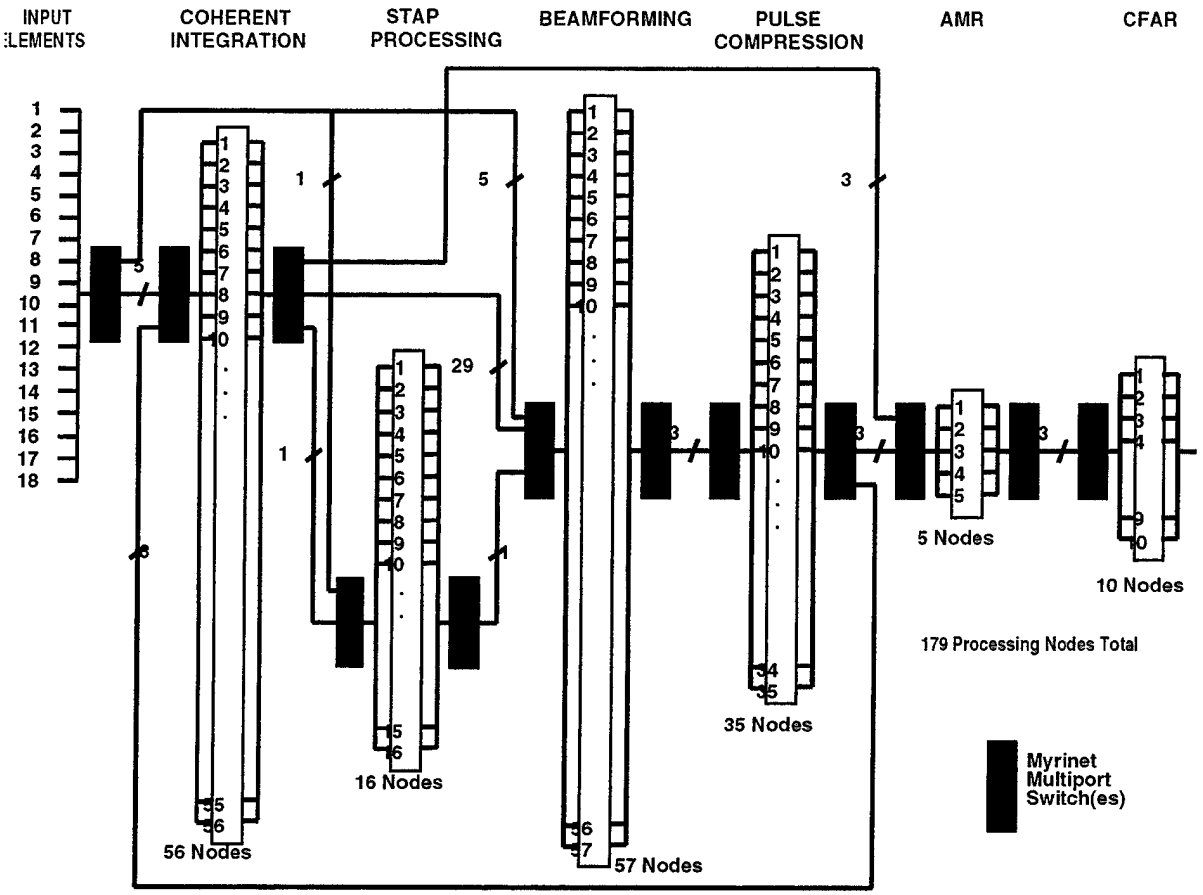


Figure 5 - Combination Pre/Post Doppler HPSC partitioning

A total of 179 processing nodes are required, partitioned into functional stages as illustrated in Figure 5 and the number of simultaneous connections between stages are indicated. The Post Doppler architecture is the driver in terms of bandwidth, requiring 29 simultaneous connections between the coherent integrator and beamformer stages. Post Doppler functionality is achieved in a sequential left to right data flow as seen in Figure 5. The Pre Doppler configuration is achieved by routing element data around the coherent integrator stage into the beamformer and then routing packet data from the pulse compressor back through the coherent integrator and on to the final processing stages.

Table 1 details the TPMs for each processing stage of the combination architecture. Included in the table is throughput in Gflops, processing efficiency (including overhead and margins), communication bandwidth (output) in MB/sec, and processing latency in PRIs.

Table 1 - TPM's for HPSC Combination Pre/Post Doppler partitioning

Function	Gflops	Proc Eff %	Comm BW	Latency (PRI)
Input data	-	-	576	-
Coherent Int	8.7	44.4	3410	1
STAP	1.9	31.3	<1	~4
Beamforming	10.3	48.6	252	1
Pulse Comp	8.8	73.1	252	18
AMR	0.6	30.0	252	1
CFAR	1.6	24.8	<40	1

For this study, a chassis is considered utilizing 12 APU and 4 XBAR boards totaling 384 processors and 60 external Myrinet™ I/O connections with an estimated 3 ft^3 volume and peak throughput of 46 Gflops. Two of these chassis are required to implement the combination Pre/Post Doppler architecture as depicted in Figure 6 by the shaded vertical bars.

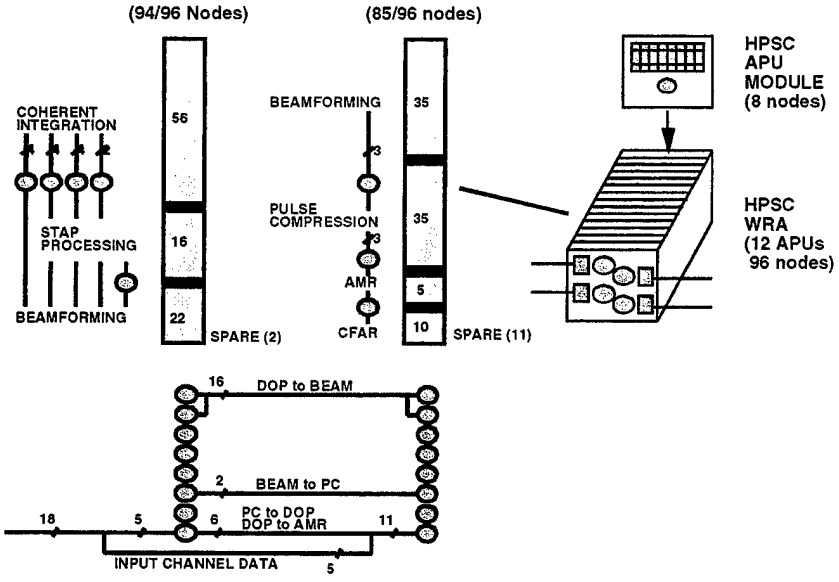


Figure 6 - Concept HPSC Chassis and Combination Pre/Post Doppler Architecture

The numbers of processing nodes allocated in each chassis for the processing functions is also shown, along with the Myrinet™ connections between APU boards. The numbers of simultaneous connections are labeled and the shaded circles represent crossbar switches on the XBAR boards as shown in Figure 4.

6.1.1 HPSC Adaptive Array Processing Modeling and Simulation

The modeling and simulation task was performed on a portion of the overall processing architecture, namely the Recursive Modified Gram-Schmidt (RMGSEF) QR Decomposition function from the Adaptive Array Processing, and performed a detailed timeline simulation to examine processor utilization and latency. This function was chosen because of the processing complexity and the requirement for a low latency solution. Because of the low latency requirement of 186 - 54x57 QR decompositions and three back-substitution solutions in 18 ms, the problem must be partitioned over a number of parallel HPSC processors that communicate through Myrinet™ and possibly additional SHARC link channels.

The RMGSEF QRD processing is modeled using a Rapid Application Specific Signal Processing [6] (RASSP) discrete event timeline simulator tool (developed by Carl Hein) called CSIM. The CSIM environment allows behavioral models to be created for each hardware element in the HPSC processing architecture. In addition, a network interconnection topology is defined between each of the hardware elements. An application software generator is then written which defines the processing tasks to be performed on each processing element (PE) using the previously created hardware and topology descriptions. The definitions are in the form of processing delays, communication packet sizes and destinations, and the appropriate timeline sequencing of interdependent events. The generator creates a program for each PE describing the required sequence of processing and communication events. The CSIM environment then combines the hardware models with the topology and application software models and performs a timeline simulation as depicted in Fig. 7.

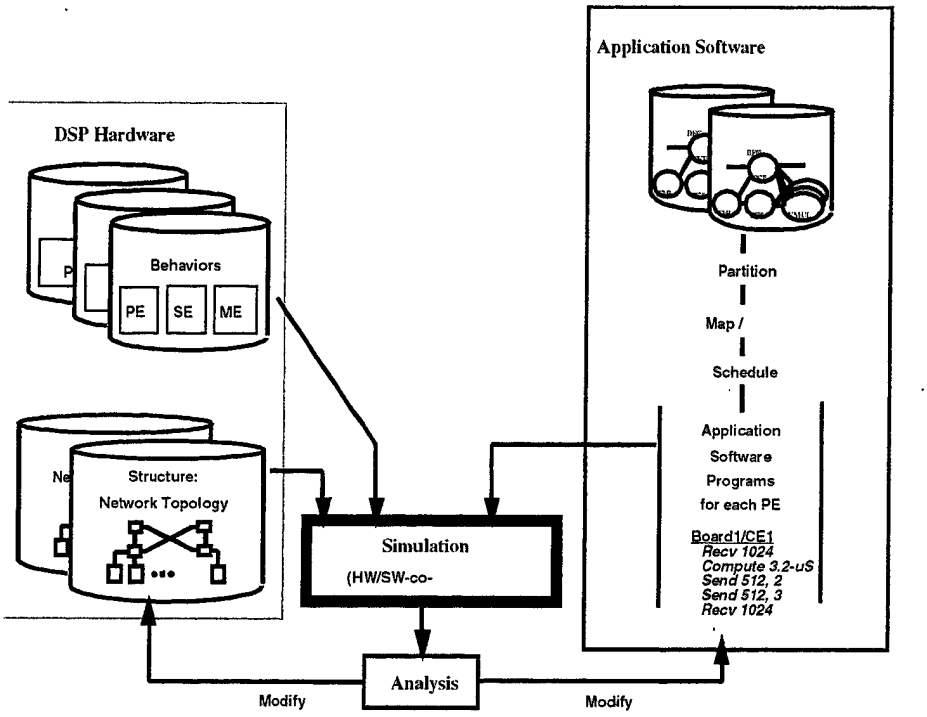


Figure 7 - CSIM Block Diagram

The outputs of this simulation may then be examined and plotted for individual processor utilization, communication network utilization and the resulting latencies for the indicated processing functions. The RMGSEF QRD algorithm performs a QR-decomposition on an upper triangular matrix of data. For the applications considered here, an input data set of 3 PRIs of 18 element data results in a 54x54 triangular matrix. This matrix has three augmented columns since weight solutions for Sum, Delta, and Omni channels are required, resulting in a 54x57 matrix. The RMGSEF data flow is illustrated in Fig. 8.

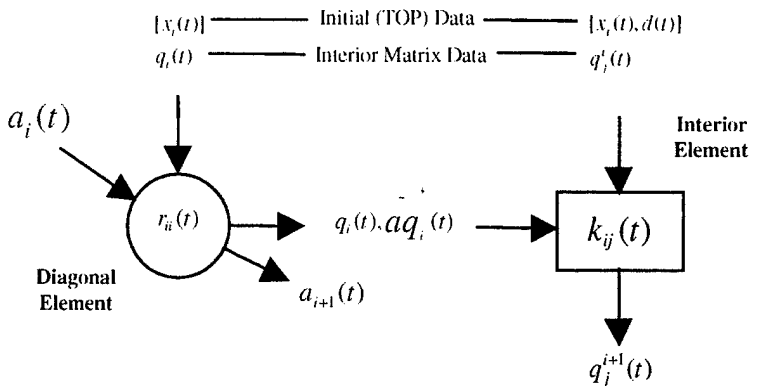


Figure 8 - RMGSEF Data Flow Diagram

Depicted are the data inputs and outputs for diagonal computation elements and interior computation elements of the matrix. Input data vectors of 57 elements, $[x_i(t), d(t)]$, arrive at the top of the matrix and are processed resulting in a set of 1593 k_{ij} terms, which are used in back substitution operations to solve for the 162 adaptive weights. Depending on the desired PRF, a range of 106 to 186 input sample vectors are processed for each QR-decomposition.

For this analysis, communication models were constructed of the HPSC SHARC PEs, the local bus interconnecting the PEs within a node, the LANai interface, and the Myrinet™ switches. Next, the RMGSEF algorithm was partitioned and mapped onto the proposed HPSC architecture in three configurations: block recursive using Myrinet™ only, sample recursive using Myrinet™ only, and sample recursive using SHARC links for matrix communications. These mappings were translated to application software processing and communication models for CSIM. Processing time estimates were created as described in the first mapping task using SHARC instruction cycle counts for the desired processing functions. Improved processing time estimates were obtained using benchmarks of optimized code for single diagonal and interior elements running on actual SHARC processors. These estimates were coded into CSIM processing delay instructions. Data packet sizes and routings were created for the proposed algorithm mappings and coded into CSIM data communication instructions.

Three different processor mappings are considered in the simulation study. A target HPSC hardware subset of 16 processing nodes is allocated for this processing task. The first mapping implements a block recursive version of the RMGSEF algorithm in which all of the 106 to 186 input sample vectors are available simultaneously as a batch for concurrent processing. This mapping utilized a vertical partitioning scheme of the QR matrix elements into the 72 SHARC PEs to maximize the vector utilization of the data within the individual PEs.

The second mapping strategy uses a sample recursive version of the RMGSEF algorithm in which input samples becomes available at periodic intervals to the top of the QR processing matrix. For this mapping, a horizontal PE allocation was used following the sequential data flow down through the QR triangle. Individual PEs do not have sufficient resources to process full rows at the top of the QR matrix, therefore, the top rows of the triangle are split into two PEs. Processing tasks are overlapped with bottom rows of the triangle which have fewer elements to perform load balancing within the PEs. This sample recursive mapping is a finer grain approach than the block recursive mapping since the processing tasks are performed on a sample basis.

These first two mappings, however, involve relatively coarse grain message transfers between processors, occurring only after entire columns or rows of processing are completed. On the order of a few hundred or thousand messages are used to simulate one cycle of the QR decomposition.

The third mapping also utilizes a sample recursive strategy, however, communications of data between individual QR matrix elements are allowed to occur using the SHARC link ports between PEs. In this model, data messages are sent horizontally and vertically as soon as they become available at an individual matrix element, rather than after a complete row has finished processing. This mapping also employs split rows at the top of the QR matrix, however, in this scheme, more sequential rows are grouped into single PEs near the bottom of the triangle to aid in load balancing. Figure 9 illustrates the QR matrix mapping onto the subset of PEs for the sample recursive with SHARC links case; each circle represents one of the two types of computation tasks as defined in Fig. 8.

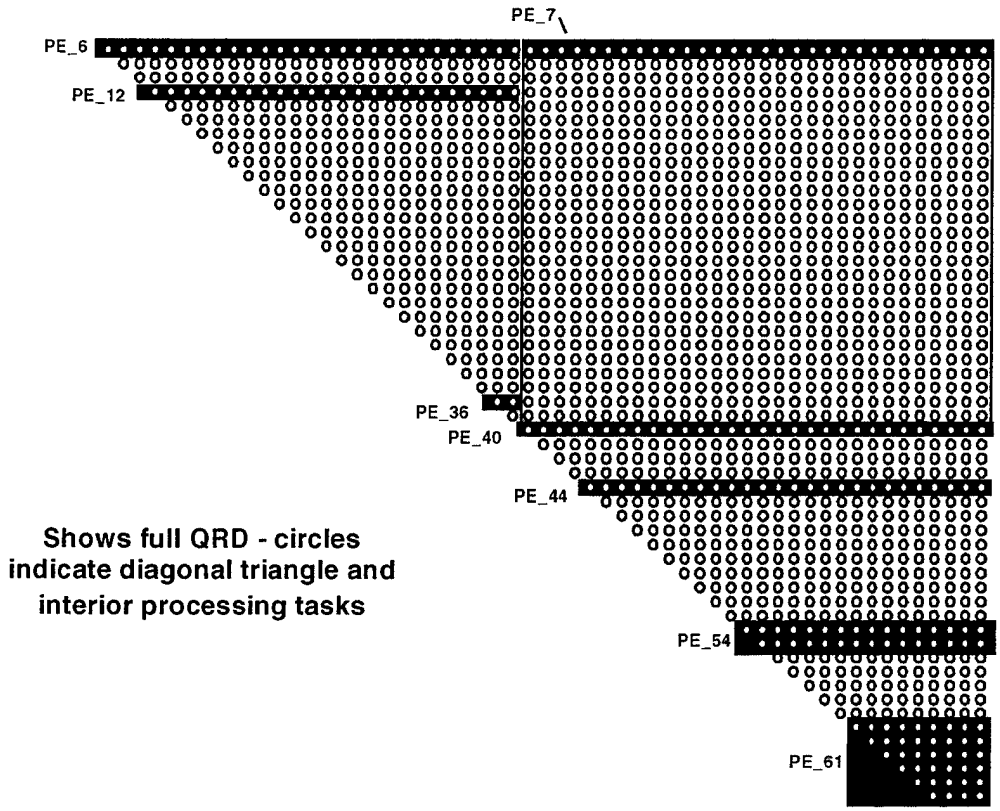


Figure 9 - Block recursive Processing Timeline

PE numbering starts at PE #6, accounting for processors which are reserved at the top and bottom of the triangle for pre and post-processing functions. This processing scheme is much finer grain than the previous two in terms of data flow, since each data message is sent to adjoining PE's as it becomes available. On the order of 500,000 messages were used to simulate one cycle of QR decomposition.

The simulation was performed over one or more cycles of QR decomposition processing and the results were plotted on a timeline as a function of the individual processors from which the overall algorithm latency is measured. Figure 10 illustrates an example simulation output depicting the processing timeline for one parametric run of five cycles of the RMGSEF QRD algorithm.

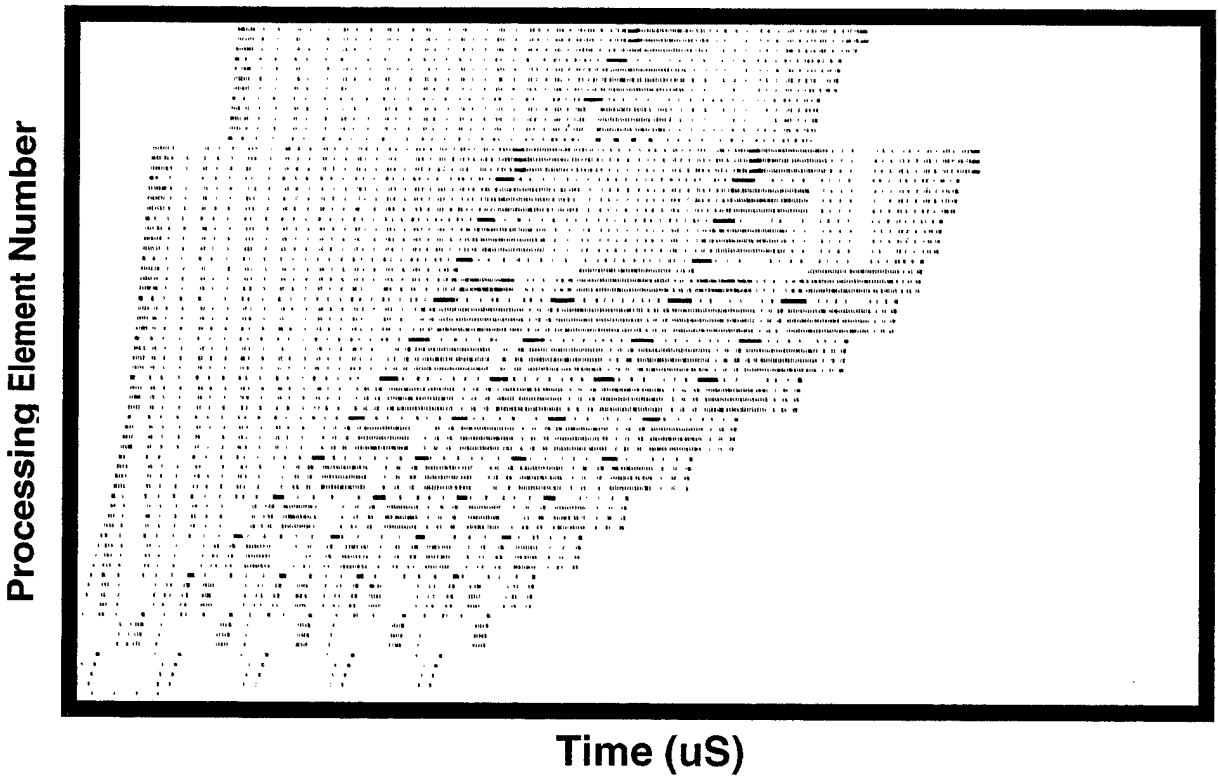


Figure 10 - Block Recursive Processing Timeline

While the HPSC hardware was under development; the CSIM environment allowed the RMGSEF QRD to be simulated as a parametric function of various processing and network time delays and efficiencies in order to evaluate the effects on the processing latency. Results are included as a function of various processing overheads and Myrinet™ transfer latency times. Figure 11 illustrates the RMGSEF algorithm latency for a specified combination of Myrinet™ and SHARC link parameters for the three mappings: block recursive and sample recursive (SR) using Myrinet™ only, and sample recursive using SHARC links. For each mapping, the latency is shown for three specified PRFs, which require processing of different numbers of sample vectors.

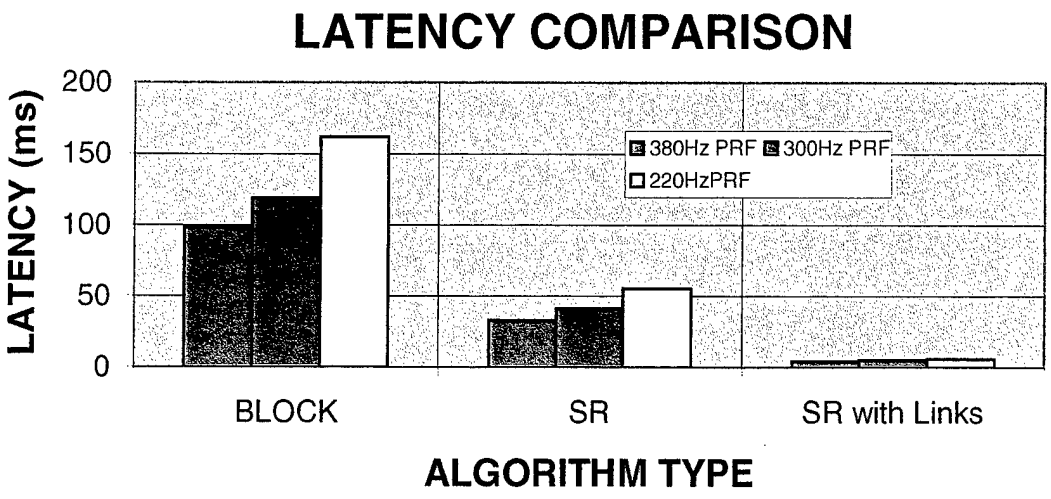


Figure 11 - Processing Latency Comparision

6.1.2 HPSC Adaptive Array Processing Demonstration

The demonstration effort focused on meeting firm requirements including real world design considerations including design margins, control and built-in-test. A set of hardware was developed and software tools were evaluated over the course of the demonstration. In addition, sets of benchmark metrics were developed from the results.

The design included meeting the following key requirements:

- RMGSEF algorithm
- 54 degrees of freedom with 3 desired responses
- Full-floating point arithmetic
- Sample rate of 41.667Khz (1 sample every 24us)
- Maximum latency of 18ms (15ms from last sample input)

In addition, real system issues were addressed including staggered pulse repetition frequencies, E-Scan Update, built-in-test, and algorithm parameter control.

The system design was performed with a goal of a minimum of 20% processing and communication margin. It was assumed that not all overheads could be accounted for in the system design phase prior to code, test and integration.

Early in the program a decision regarding the processor was required in order to meet the lead times associated with procuring and developing hardware. The ADSP21062 was selected based on its relative maturity and availability as compared to the ADSP21060. The memory limitations of the ADSP21062 were considered lower risk than the anomalies associated with the ADSP21060. Development of the first Arithmetic Processing Unit indicated that the 40MHz processing cycle was not attainable due to loading on the Myrinet™ LANai chip. Thus the module operating frequency and therefore the processor frequency was downgraded to 33MHz (32.51MHz to be more precise). This translated into a maximum of 792 instruction cycles in a 24ms sample interval. Incorporating the design margin resulted in a maximum of 634 cycles in a 24ms sample interval. The SHARC Link communication was also adjusted with the partitioning requiring only 80% of 1X link port speeds. This resulted in 13Mb/s peak transfer rate on the SHARC Link or 317Bytes every 24ms sample interval.

Based on the above processing limitations and baseline partitioning, it was determined that 14 nodes would be required to source intermediate calculations relating to weight back-substitution. The worst case update rate is 380Hz or every 2.6ms. This establishes 1376 Bytes as the worst case number of data bytes output by a processing node. Thus the Myrinet™ performance

requirement was to transfer 1376Bytes in (80% x 2.6mSeconds) / 14nodes or 1376Bytes in 148mSeconds. (The best case transfer was 664Bytes and the average transfer was 910Bytes).

The diagonal element of the RMGSEF algorithm was optimized to perform in 27 cycles. The internal elements of the QRD were optimized to 9 cycles each. The processing overhead was broken down into control, subroutine calls and buffer polling. This overhead processing was estimated to be 100 cycles, resulting in 534 cycles (634 - 100) for diagonal and internal element algorithm processing.

The system hardware was designated as the Airborne Early Warning Scalable Computer (AEWSC). It consisted of a single enclosure and chassis containing cooling fans, power supplies, a seven slot VME backpanel and a thirteen (13) slot HPSC backpanel. The VME backpanel was populated with an embedded SPARC-10 single board computer from Force computers, two (2) custom interface cards to exchange data with the Digital Beamformer (DBF) and a MMV card from Sanders containing nine (9) ADSP21062s operating at 32.5MHz. The HPSC backpanel contained eight Discrete Arithmetic Processing Units (DAPUs) and a single Memory Topology Expansion Module (MTEM). A block diagram of the AEWSC is illustrated in Figure 12.

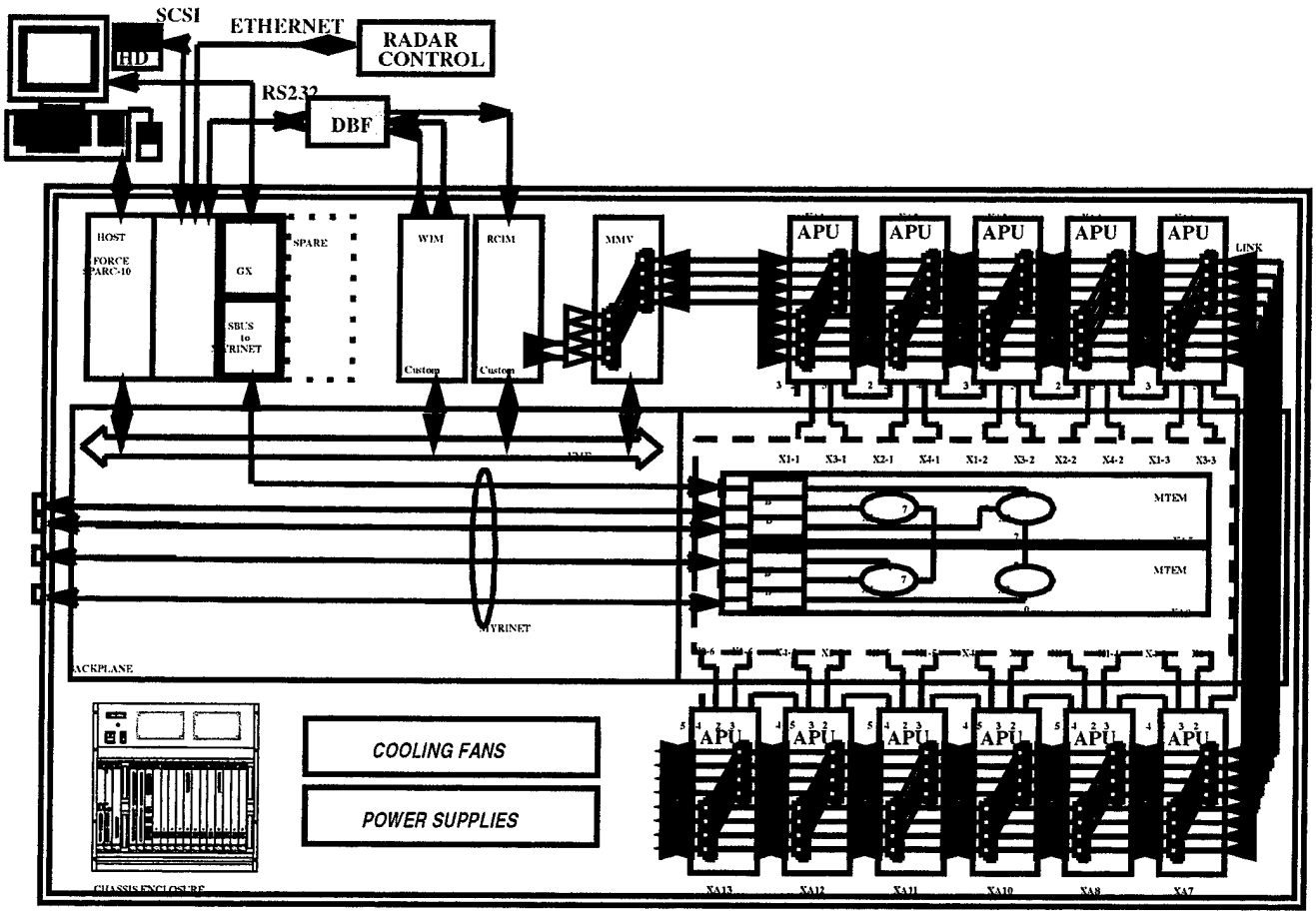


Figure 12 - AEW Scalable Computer - Adaptive Array Processor

The Discrete Arithmetic Processing Unit (DAPU) contains two processing nodes where each node consists of four ADSP21062 digital signal processors operating at 32.5MHz. Each node supports up to 64MBytes of DRAM and accesses the Myrinet™ network via a dedicated LANai processor with 1MByte of SRAM. The nodes or the DAPU are connected via a combination of SHARC Links and an 8-port Myrinet™ switch. The module supports four (4) Myrinet™ bidirectional links to the HPSC backpanel and a total of sixteen SHARC Links for interconnecting adjacent modules. See Figure 13 for a block diagram.

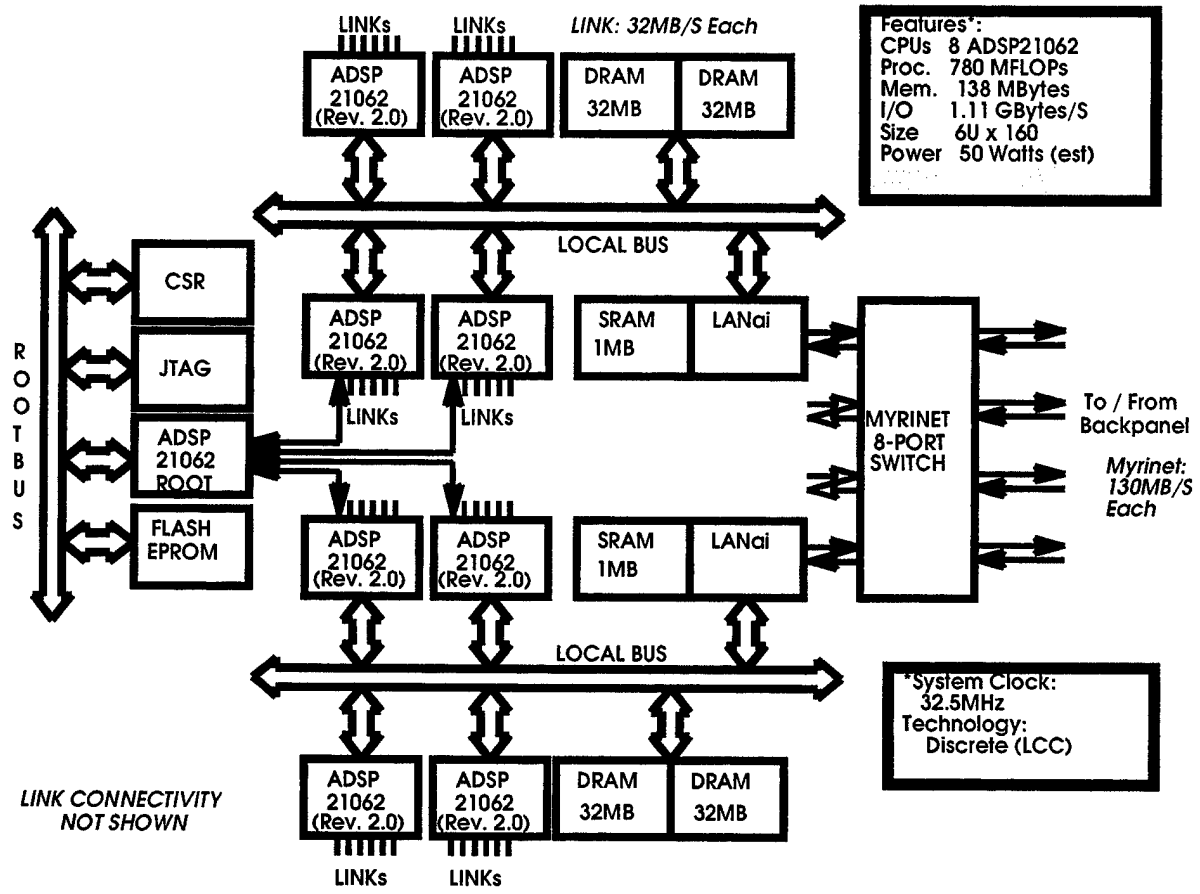


Figure 13 - HPSC Discrete Arthmetic Processing Unit (DAPU)

The Myrinet™ Topology Expansion Module (MTEM) contains two Myrinet™ 8-port switches and provides for module-to-module and chassis-to-chassis communication. Each module supports thirteen (13) backpanel Myrinet™ links (1-meter protocol) and three chassis-to-chassis links (25-meter protocol). The MTEM provides a two-dimensional mesh packet switching network with flow control and cut-through routing associated with Myrinet™. See Figure 14 for a block diagram.

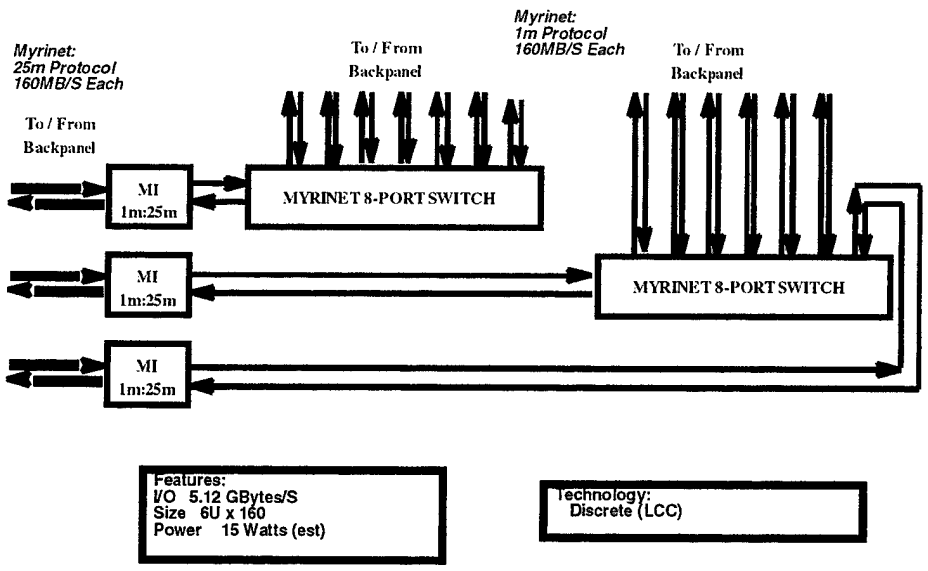


Figure 14 - HPSC Myrinet™ Topology Expansion Module (MTEM)

The software environment employed to develop the adaptive computation and STAP control functions is outlined below:

- Host / STAP Controller Development Tools
  - Force SPARC-10 Solaris 2.4
  - GNU C Developers Tool Kit
  - VME and RS232 Drivers
  - Myrinet™ Software (Myrinet™ / SBus-Solaris 2.4)
  - Motif Graphical User Interface
  - Visual Data Analysis Software
- HPSC Development Tools
  - ADI Tools (Version 3.2)
  - Myrinet™ Communication Library
  - SHARC Libraries
  - SBUS Myrinet™ Driver
- HPSC Runtime Tools
  - Debug Monitor (Version 1.0 Release 96.11)
  - SHARC and Myrinet™ Profilers
  - File Conversion Utilities

Note that no real time operating system was used in the development of this application.

The STAP / HPSC software was subdivided into two: STAP Controller supporting operator control of operation and test, beamsteering processing and weights display, and the Adaptive Array Processing (AAP) consisting of QR decomposition of sampled data matrix and explicit evaluation of weights via back-substitution. Additional AAP consisted of screening of sample data, format conversions between fixed and floating point and summation with the beamsteering data.

The AAP code was further partitioned into Summation, QRD, and Weight Back-substitution. The summation code resulted in four executable code segments running on seven processors. The QRD code was reduced to a single executable code segment running on fifty-six processors. The Weight Back-substitution code consisted of four executable code segments running on four (4) processors.

The user interface to the STAP Controller consisted of a Motif Graphical User Interface (GUI). The GUI provided for a hierarchic set of windows providing prompts and data entry windows to the user. The STAP Control software was partitioned according to these user interface windows consisting of:

- File
- DBF (Digital Beamforming)
- BSP (Beamsteering Processing)
- AAP (Adaptive Array Processing)
- RTP (Real-Time Plotting)
- HELP (On Line Help)

In all, 43 user screens were developed to provide the user with control and test of the entire STAP subsystem.

6.1.3 Bechmark Metrics

The measured benchmarks are listed in Table 2.

Table 2 - Benchmark metrics

	QRD	*Weight*	*NOTE*
Sustained Input Data Rate	20 MB/S	5 MB/S	Bytes of data input per unit time
Sustained Computation Throughput	1,111.5 MFLOPs/S	13.2 MFLOPs/S	Achieved results per unit time
Peak Processing	56 Processors 3640 MFLOPs	1 Processor 65 MFLOPs	# Processors x 32.5 MHz x 2 FLOPs
Overall Efficiency	30.6%	20.3%	Sustained / Peak
Fixed Workload Speedup	182	N/A (Single Processor)	Ratio of time for a Single processor / Multiple Processor
Computation-to-Communication	1	2.7	16FLOPs/16Bytes, 34344FLOPs/12,744Bytes
Local Memory Demand	20%	23%	ADSP21062 = 65,536 Words
Execution Time (Latency)	< 7 milliseconds		Measured wall time from first sample input to weight output

Execution time or latency was a key driver in the mapping of the application to the architecture. The maximum latency required must be less than 15 milliseconds from the time the last sample arrived until the last weight was computed and ready to be transferred to the DBF.

The Sustained Input Data Rate is 20 MB/S for the QRD portion of the processing function. The internal QRD processing was performed in 39 pipelined processing stages. Transfers internal to the QRD processing varied with an average input data rate of 12.8MBytes/Second, and the average output data rate of 12.3MBytes/Second at each of the 39 stages.

The Sustained Computation Throughput was calculated for the QRD and Weight processing operations. The QRD processing required an inverse operation that was estimated at 12 FLOPs. Thus 135 samples times 26,676 FLOPs (=3.6GFLOPs) are performed in 3.24microseconds resulting in 1,111.5MFLOPs/Second. The screening and format conversion were not included in this benchmark.

Peak Processing of a single processor was determined by multiplying the processor clock rate by the number of operations applicable to the problem, which in this case were two: multiply and add. (FFTs were not being performed thus the third simultaneous operation could not be utilized.)

The Fixed Workload Speedup is the ratio of the time needed for one processor to perform a fixed task to the time needed by the multiple processor. The Fixed Workload Speedup was obtained in three of steps. The first step involved optimizing the QRD code. The non-optimized QRD code on a single ADSP21062 required 591mSeconds to process 135 samples. After using Standard optimization (leveraging both Program Memory and Data Memory and using compiler switch -O3) the QRD code on a single ADSP21062 required 362ms to process 135 samples. The second step involved development of a custom library routine optimized to perform the QRD. This effort resulted in a single ADSP21062 requiring 67.4ms to process 135 samples. The final speedup was obtained by distributing the code over 56 processors resulting in a time of 5.112ms to process 135 samples. (The times are based on a 32.5MHz clock cycle). See figure 15 for a block diagram depicting this process.

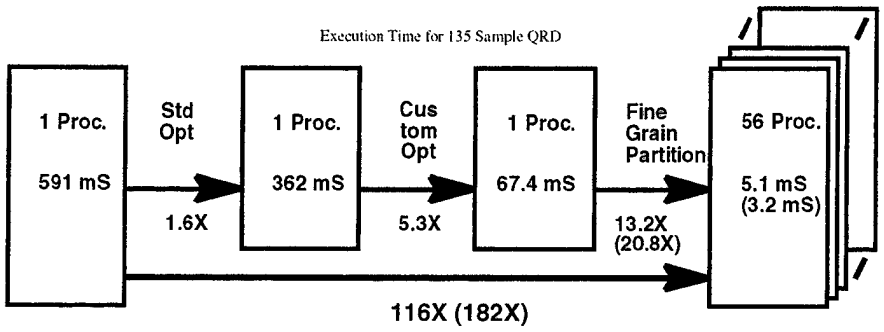


Figure 15 - Fixed Workload Speed Up

Note that the 5.1ms time includes the QRD pipeline delay and illustrates the latency. The QRD was performed on 135 samples every 3.24ms across the 56 processors resulting in a 21X speedup. In other words, 1.9ms (5.1ms - 3.2ms) is required to fill and empty the QRD processing pipeline. The overall computation speedup resulting from the combined effort of software optimization and multiprocessing was 182X.

The Computation-to-Communication for the QRD is based on the Internal Cell computation that is responsible for 95% of the QRD computation. The total computation within the cell is 16 FLOPs. Each internal cell receives a single complex element made up of 8 bytes and outputs a single complex element made of 8 bytes.

The Local Memory Demand was not a driver given that the algorithm was so finely partitioned over many processors. The resulting load image was identical for all processors performing the QRD.

6.1.4 AEWSC Components

Figures 16 through 21 are pictures of the components utilized in the AEWSC.

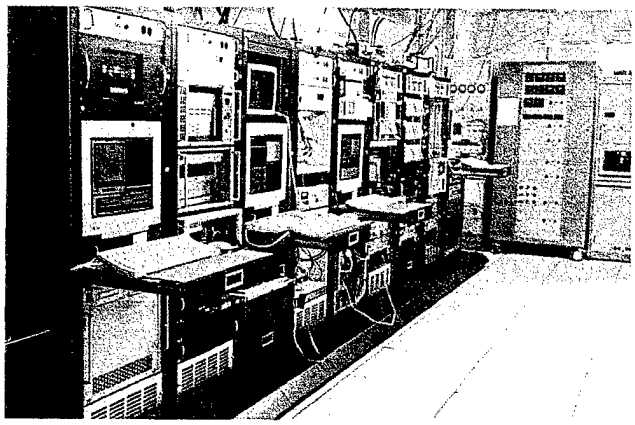


Figure 16 - HPSC Lab Configuration

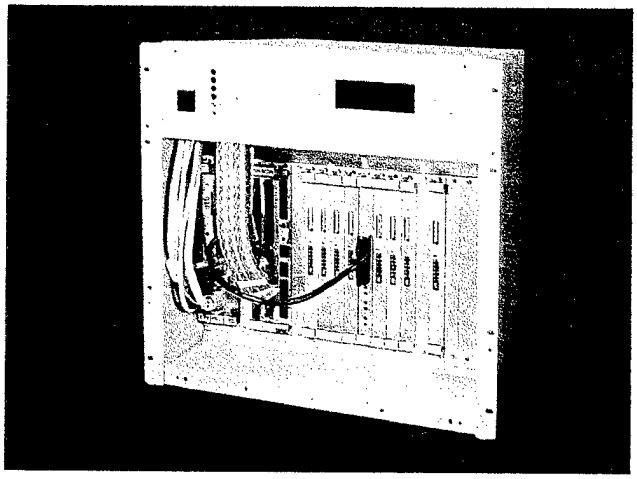


Figure 17 - AEWSC Chassis

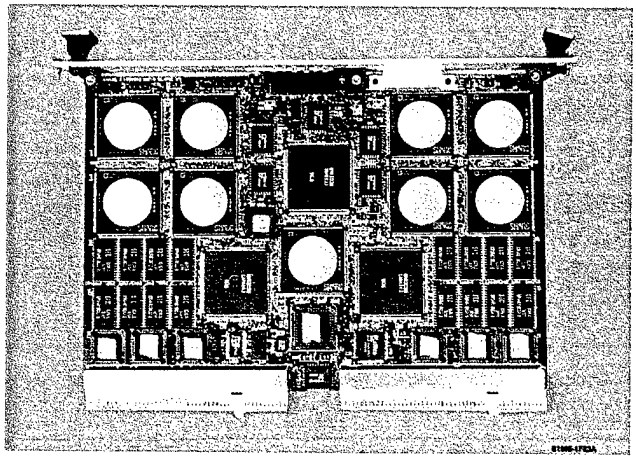


Figure 18 - HPSC APU Module

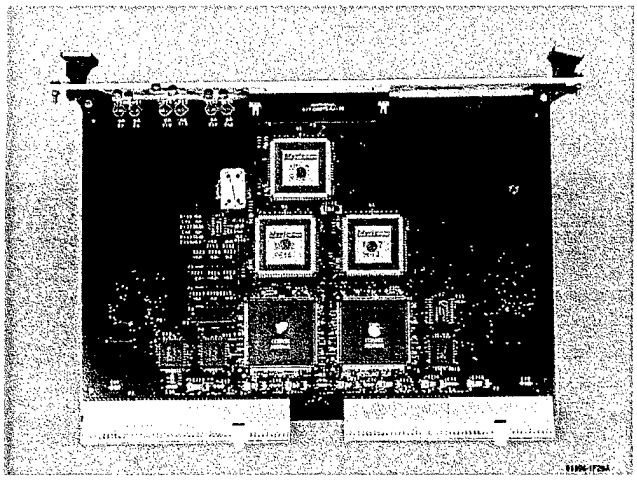


Figure 19 - HPSC MTEM Module

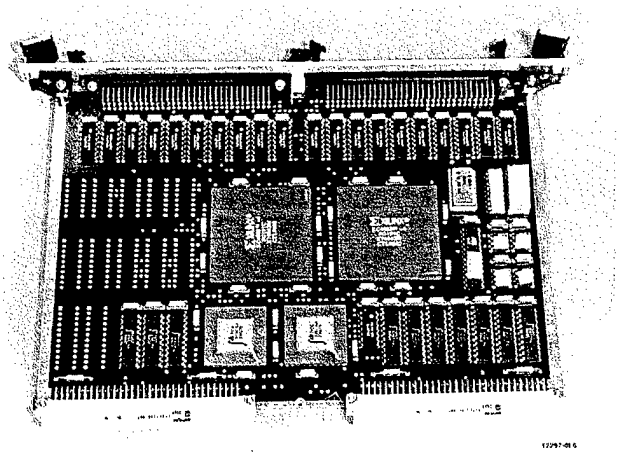


Figure 20 - Weight Interface Module

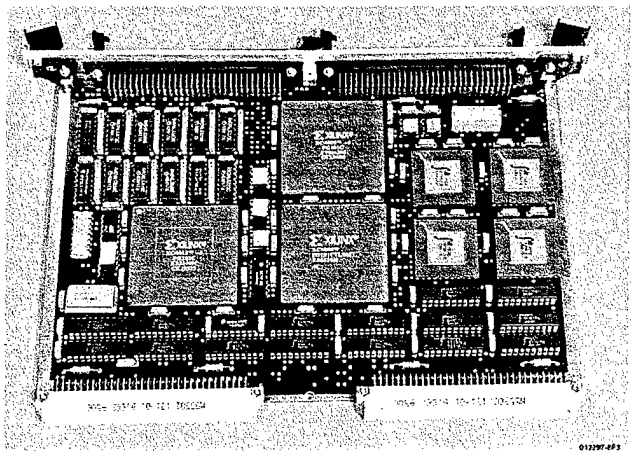


Figure 21 - Radar Input Control Interface Module

6.2 Adaptive Array Processing on COTS Platforms

6.2.1 Introduction

The purpose of benchmarking the RMGSEF algorithm on COTS platforms was to determine the processing latency and minimum hardware requirements to implement the RMGSEF algorithm in the allotted time. Two benchmark programs were developed. The target processor for this trade study was a PowerPC (PPC) microprocessor.

This report contains the following sections:

- Description of algorithm to benchmark
- Measurements required
- Benchmark Implementation steps
- Analysis/Summary of results

6.2.2 Algorithm to Benchmark:

The benchmark programs were implemented using the PPC on VME modules supplied by the vendors CSPI and Mercury Computer Systems. The major implementation differences between the vendor modules are found in the board interfaces. CSPI uses a Myrinet™ interface, while Mercury Computer Systems uses a Raceway interface.

The algorithm to benchmark receives, as input, 135 frames of 54 elements and 3 channels every 3.2ms. The RMGSEF algorithm receives floating point input data. The input data labeled Rdata in Figure 22, received from RCIM is summed with data from the BSP table. Refer to Figure 22 for a description of this process.

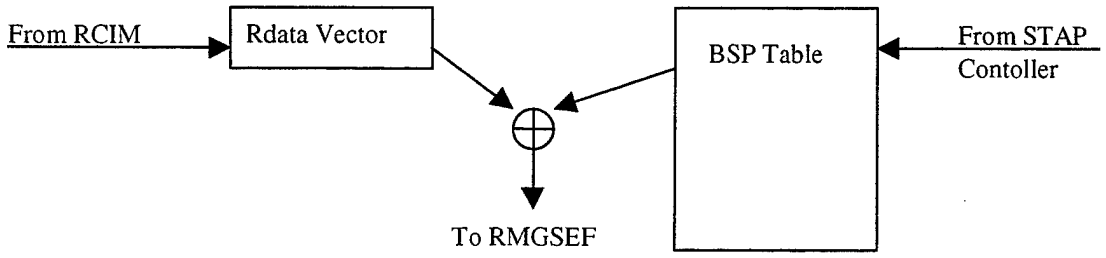


Figure 22 - Data Flow for Summation Process

After the summation process is complete, the data passes through the QRD processors. The QRD function is the matrix computation used to calculate the 'K' values. Refer to figures 23 and 24 for the RMGSEF algorithm data flow.

After processing all frames of data in a PRI, the K's are then compressed in a back substitution method to create the weights. These weights are then output to the Digital Beamformer for use in forming the beams. Refer to Figure 25 for the weight generation from the K's generated in the RMGSEF algorithm.

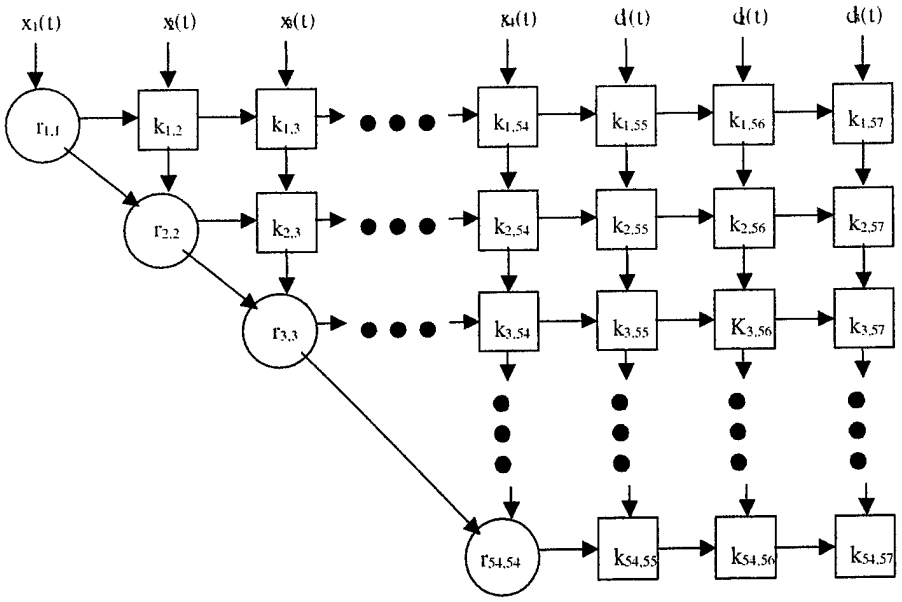


Figure 23 - Data Flow for RMGSSEF Algorithm

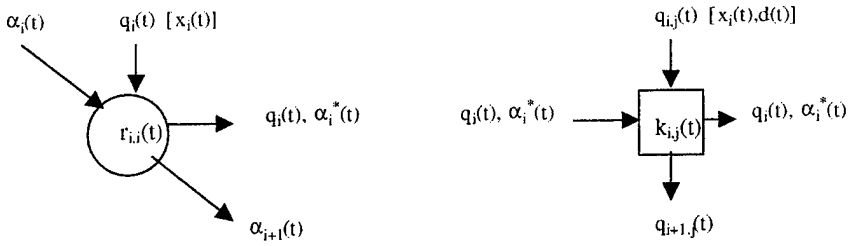


Figure 24 - Single Node Expansion in RMGSEF Algorithm

$$W_i = kd_i - \Sigma W_h k_{ih} \; ; \; i+1 =< h =< 54 \; ; \; 1 =< i < 54$$

Figure 25 - Weight Generation

6.2.3 Performance Measurements

To accurately evaluate the performance of the RMGSEF algorithm on the PPC, measurements need to be recorded for each benchmark. The following are the measurements that were compiled.

The Algorithm measurements include:

- Process time for each sample through summation processing.
- Process time for each row or QRD matrix.

- Process time of entire matrix per sample.
- Process time to calculate weights from matrix.
- Time to process all samples.
- Overhead reduced by blocking samples together.
- Latency through system.

6.2.4 Measurement of Processing Latency

Processing Latency through the system is defined as time from which the first input frame is received to the time the calculated weights are output for use. Multiple steps were taken to determine the number of processors required to compute the algorithm at the specified data rate and latency for the data to progress from start to finish.

Each benchmark was implemented in three steps. The first step measures the performance utilizing a single processor. The second step utilizes four processors and the third step on more than four processors.

6.2.4.1 Single Processor Performance

Before deciding on any allocation of functions to processors, the RMGSEF algorithm was benchmarked on various PPC platforms. The platforms consisted of different speed processors and busses on Mercury Computer System and CSPI boards. The measurements were taken by running the function for 100 passes and the average time for each step was calculated. Tables 3 and 4 show the results of the measurements.

Table 3 - Performance Set 1

Function	Mercury 100Mhz PPC	Mercury 200Mhz PPC	Mercury 300Mhz PPC – 750 with 1MB External Cache
Sum/sample	83.322 us	45.897 us	24.593 us
QRD/sample	771.461 us	494.923 us	157.518 us
Wts Compress	808.620 us	433.740 us	214.848 us
Total Latency	119.086 ms	76.120 ms	25.388 ms

Table 4 - Performance Set 2

Function	CSPI 200Mhz PPC w/ 33Mhz bus	CSPI 200Mhz PPC w/ 66Mhz bus	CSPI 300Mhz PPC w/ 66Mhz bus
Sum/sample	54.0 us	43.0 us	34us
QRD/sample	629.0 us	335.0 us	328us
Wts Compress	490.0 us	377.0 us	291us
Total Latency	96.939 ms	52.548 ms	51.1ms

After collecting the times for all functions, the implementation of the summation function was found to be inefficient. The function was receiving data as packed complex fixed point. This data was then unpacked, floated, and then summed. An assumption made here is that all data is received by the summation function as full 32-bit floating point. The summation function was modified and the test re-run on the 200Mhz PPC (Mercury board). Using the new implementation the timing of the summation function (9.3 us) showed a 5x improvement in speed.

6.2.4.2 Four Processor Performance

Based on the measurements of each of the functions shown above, the following shows the functional allocation to processors for the full up multi-processor architecture. Each solid lined box in Figure 26 represents a PPC.

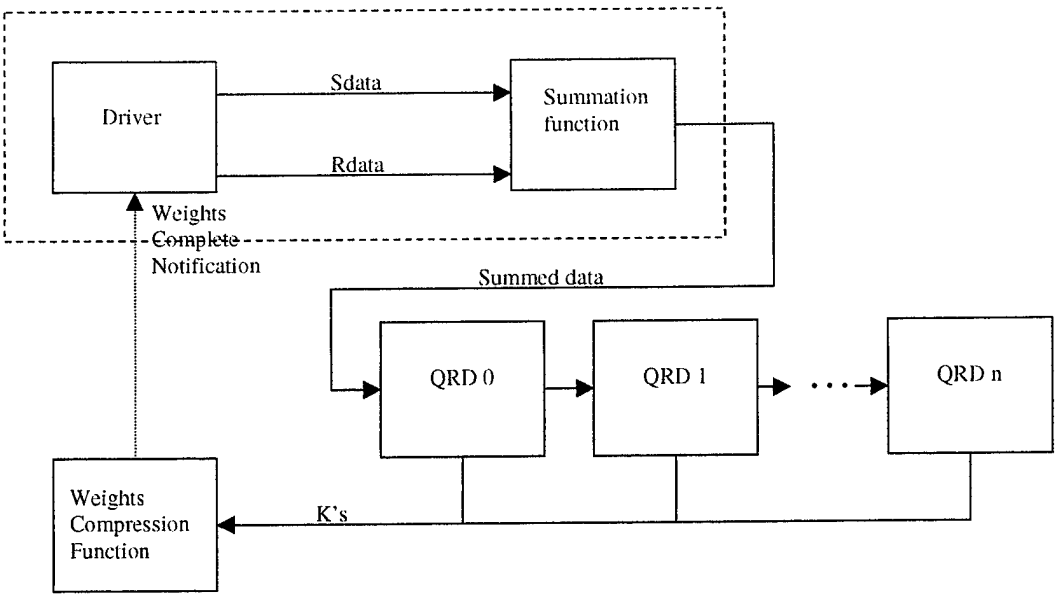


Figure 26 - Multi Processor Block Diagram

The solid lines connecting the processors represent the interface (Myrinet™ – CSPI/ Raceway – Mercury). For this study only four processors were utilized due to limited hardware resources. Because of this, the summation function was combined with the data driver image to allow it to be executed by the available processors. In addition, step 2 only used 2 processors to run the QRD function. All of the interfaces were implemented and verified and timing measurements will be taken to determine overhead savings and latency.

For step 3 (full up architecture) additional processors will be utilized so that the RMGSEF algorithm can be executed in the required time. At that time the full latency can be measured and compared between platforms.

6.2.4.2.1 Data Flow and Processor Description

6.2.4.2.1.1 Data Driver

The flow of data through the processors starts at the data driver. The data driver reads in data from files containing the Sdata and Rdata. This data is floated and setup for transmission to the summation function. Parameters input to the data driver include:

- Sdata filename
- Rdata filename
- Timing results filename
- Number of samples to process
- Number of samples blocked together
- Number of passes (i.e. number of time to calculate weights)

After formatting the data for the test, blocks of data are output to the summation function. Each time a sample (or block of samples) is output to the summation function a time stamp is recorded. The time stamp is compared to the time stamp taken when a weights complete notification is received. The data driver continues to output data until all passes (command argument) have been completed. At this time a "halt" command is sent through the pipeline of processors. The recorded time stamps are then output to a results file for inspection.

6.2.4.2.1.2 Summation Function

Upon receiving data, the summation function records a time stamp. The data is then processed and sent to the first QRD processor. After the data transmission is started a second time stamp is recorded. These time stamps are used to calculate the total time it took to process the block of data input. Parameters input to the summation function are:

- Timing results filename

The summation processor continues to process data until the "halt" command is received. At this time the recorded time stamps are output to a results file for inspection.

6.2.4.2.1.3 QRD Function

Upon receiving data, the QRD function records a time stamp. If a "reset" command is received, the message is passed along to the next QRD processor and the intermediate values are reset. If a "compute QRD" command is received, the QRD function is processed and then the results are sent to the next QRD processor. If a "collect K" command is received, all the K's are output to the weight compress function. After the block of data is processed a second time stamp is recorded. These time stamps are used to calculate the total time necessary to process the block of input data. Parameters input to the QRD function are:

- Start row to process
- Number of rows to process
- Timing results filename

The QRD processor(s) continue to process data until the "halt" command is received. At this time the recorded time stamps are output to a results file.

6.2.4.2.1.4 Weight Compression Function

Upon receiving data, the weight compression function records a time stamp. When all the K's are received from all the QRD processors, the weights are calculated by compressing the K's matrix. After the weights are calculated, a notification is sent to the data driver. For the Mercury benchmark this notification is a mailbox interrupt. For the CSPI benchmark the notification is a message. The data driver then records the time the notification is received. This time is used for analysis later to determine the total latency throughout the entire function. After sending the notification, the calculated weights are compared with known good weights. Status of this comparison is recorded. Parameters input to the weight compression function are:

- Timing results filename

The weight compression function continues to process data until the "halt" command is received. At this time the comparison results and timing results are output to a results file.

6.2.4.3 Performance Results

Step 2 of this process was limited to 4 processors on each platform. The two QRD processors were setup to process about half the number of computations in the RMGSEF matrix. The first QRD processed the first 17 rows. The second processed the last 37 rows. The main objectives of this step were to measure overhead savings by blocking samples together and to prepare the code for expansion to the full set of required processors in step 3. Measurements were taken at the time of the first sample output, the start and end of each function processing data, and the time when weights were fully calculated. Table 5 shows the measurements collected for each platform.

Table 5 – Multi Processor Performance

Platform	Process Block Size	Summation Time Per Block	QRD0 Time Per Block (First 17 Rows)	QRD1 Time Per Block (Last 37 Rows)	Weight Time Per Block	Total Latency
Mercury 200Mhz	1	8,160 ns	157,925 ns	141,868 ns	466,500 ns	27,602,820 ns
Mercury 300Mhz	1		72,720 ns	70,704 ns	257,280 ns	13,264,464 ns
Mercury 200Mhz	2	16,685 ns	151,708 ns	137,808 ns	464,760 ns	25,230,180 ns
Mercury 300Mhz	2		144,000 ns	140,304 ns	257,184 ns	12,246,288 ns
Mercury 200Mhz	4	30,447 ns	589,035 ns	539,837 ns	465,300 ns	24,741,660 ns
Mercury 300Mhz	4		303,600 ns	281,520 ns	256,992 ns	11,831,664 ns
Mercury 200Mhz	8	60,013 ns	1,149,828 ns	1,063,595 ns	464,880 ns	25,152,060 ns
Mercury 300Mhz	8		569,040 ns	559,488 ns	257,088 ns	11,808,816 ns
Mercury 200Mhz	16	123,821 ns	2,143,636 ns	1,991,357 ns	465,060 ns	25,765,980 ns
Mercury 300Mhz	16		1,133,952 ns	1,120,800 ns	257,232 ns	12,206,304 ns
Mercury 200Mhz	32	332,819 ns	3,844,920 ns	3,556,872 ns	466,440 ns	28,067,580 ns
Mercury 300Mhz	32		2,260,032 ns	2,244,480 ns	258,048 ns	13,235,808 ns
Mercury 200Mhz	64	534,650 ns	6,376,880 ns	5,918,720 ns	465,900 ns	32,898,060 ns
Mercury 300Mhz	64		4,507,536 ns	4,484,928 ns	258,048 ns	15,307,536 ns
Mercury 200Mhz	135	1,620,999 ns	19,088,640 ns	17,669,100 ns	466,260 ns	61,154,100 ns
Mercury 300Mhz	135		9,500,160 ns	9,433,680 ns	257,472 ns	39,943,344 ns

Where:

Total Latency is the time from the output of the first sample to summation to the time the "Weights Complete" notification is received by the data driver.

6.2.4.4 Full-up Architecture

Due to limited availability of hardware, this step of processing the data at full speed with a full compliment of processors has not been completed at this time. Table 6 contains the estimates of the number of processors needed and the approximate latency and execution time of the algorithm. The time per computation was calculated using the QRD time on one processor and the number of non-zero cells in the RMGSEF Matrix.

Table 6 - latency Estimates for 300Mhz Mercury PPC Module

Estimated Latency
4,445,707(ns)

Sum Time per Sample	QRD Time Per Sample	Weight Time	Number of Computations in RMGS Matrix	Time Per Computation
24593ns	157518ns	214848ns	1647	114.767213ns

Interface Transfer Speed (50% derated)	Number of Samples Per Block	Time available to process number of Samples/Block
12.5 ns/Byte	4	96000

Processor number	Number of Rows Processed	Processing Time for Number of Rows	Message Size	Data Transfer Time
1	3	77123.57ns	1832	22900ns
2	3	72991.95ns	1736	21700ns
3	3	68860.33ns	1640	20500ns
4	4	86304.94ns	1544	19300ns
5	4	78959.84ns	1416	17700ns
6	5	89518.43ns	1288	16100ns
7	5	78041.7ns	1128	14100ns
8	6	79877.98ns	968	12100ns
9	8	84468.67ns	776	9700ns
10	13	89518.43ns	520	6500ns
		TOTALS		
	54	805665.8ns		160600ns

6.2.4.5 Summary

The benchmark programs implemented on the PPC were done to determine the total latency incurred when running the RMGSEF algorithm on the CSPI and Mercury boards. Using the first two steps stated above, we are able to make a better estimate on number of processors required to process data through the RMGSEF and what the expected latency would be.

The I/O load from processor to processor is minimal. Both interface types are capable of transferring up to 160MBytes/sec and all transfers are estimated to require no more than 25% of that peak. As samples are blocked together, the overhead of the transfer setup is less and therefore will reduce the I/O load even more.

One problem found with the Mercury Raceway is that the console server periodically checks up on each board in the system. When this occurs we have noticed a difference in time to complete certain functions. For example when measuring the times for each sample being processed by the summation function, we would see an occasional 40-60us measurement. The normal time measurement of this function is about 7-8us. Unfortunately this can not be turned off. If the I/O scheme is not designed properly, this added time may cause the algorithm to miss some data. If the I/O scheme is not designed to accommodate this feature, the added time may cause the algorithm to miss some data and collapse the refined solution.

Overhead savings and total latency decreases as samples are blocked together. After 8 samples the total latency starts to increase. Therefore blocking of samples should be limited to no more than 8.

As shown in Table 5, 10 - 300MHz PPC-750 processors would be required to process the data at real time. The current estimate for the same benchmark on the CSPI 300Mhz board may require more processors since much of the speed of the algorithm is limited by the cache memory available in the processor. The limitation can be seen in the timing comparisons of the 200Mhz and 300Mhz times collected on the CSPI board.

The information in Table 6 supports this estimate and even shows that fewer processors would be needed. The latency measured when using 2 PPC processors for the QRD portion of the algorithm 12.5ms.

6.3 Adaptive Array Processing on the Ixthos Platform

The Ixthos platform utilizing the IXZ16 module is evaluated as an alternative solution for the Adaptive Array Processing. A reduction in size, weight and power consumption is the goal over the HPSC solution.

A functional block diagram of the AAP utilizing the HPSC modules is given in Figure 12. The alternate AAP architecture utilizing the IXZ16 is shown in Figure 27.

The functions of the MMV, such as data screening and format conversion, as well as weight calculations performed by APU boards are now handled by one IXZ16 board. In addition, the QR decomposition requires four IXZ16 boards as opposed to eight boards in the HPSC architecture.

Each IXZ16 board has link port support in addition to VME and an optional IXI32D multi-drop interfaces. The link ports allow communication from processor-to-processor, node-to-node, and board-to-board similar to the HPSC.

The QRD processing would be performed in a similar manner as is performed in the HPSC with the transfer of K's taking place over the IXI32D multi-drop interface.

6.3.1 The Ixthos AAP Data Path

The data flow and control remains largely the same in this architecture as in the HPSC architecture. The main difference is the absence of the Myrinet™ interface. The functions of the RCIM and the STAP controller are explained in the following paragraphs which were taken from the AEW Scalable Computer Beam Steering Processor Baseline System Definition, revision I.

Figure 27 shows the data flow between the hardware components of the AEWSC. This provides an overview of the AEWSC, and shows the partitioning between the hardware and the software. It also illustrates the interfaces between the various modules particularly the interface between the hardware and software.

The STAP Controller module acts as the VME slot 1 controller. It interfaces to a User Interface in the form of a monitor, keyboard and mouse. The STAP Controller also accepts radar commands, including operating frequency and electronic steering angle, from the Radar Controller. The STAP Controller distributes radar commands and user commands to the appropriate modules over the VMEbus. The STAP Controller queries the various modules on the VMEbus for status. This module is an embedded SPARC–10 board from Force computers with a VME interface with master capability and a Myrinet™ interface.

The STAP Controller receives user supplied Mode information and Commands and relays this to the AAP and DIU. The Modes consist of BIT Mode, Adaptive Mode with Periodic BIT, Adaptive Mode without Periodic BIT, Reset Adaptive with Periodic BIT and Reset Adaptive without Periodic BIT.

Commands supplied by the STAP Controller consist of Initialization commands, Operation commands and an algorithm Reset command with the exception of the Initialization command which initializes the entire AEWSC, commands are issued to the DIU - RCIM module over the VME bus.

The Parameters are RMGSEF algorithm parameters consisting of RMGSEF - Alpha, Gamma, QRD:PRI and Exponents for RDATA, and Weight data format conversions. Alpha and Gamma are values used directly in the RMGSEF algorithm. The QRD:PRI parameter forces a certain number of QRDs to be used in updating the weights. The exponents are used to convert the RDATA from 16–bit fixed point values to 32–bit full floating point values, and the Weights from 32–bit full floating point values to 16–bit fixed point values. Alpha, Gamma and Exponents are supplied to processors responsible for format conversion over the Myrinet™. The QRD:PRI is supplied to the RCIM so that proper control may be generated.

The User may request Monitor and Status information to be gathered and displayed by the STAP Controller. The Monitor words indicate the current operation of the AAP and DIU. The Status Words indicate the occurrence of certain events in the AAP and DIU. The STAP Controller must query the various AAP / DIU modules over the VMEbus to determine the complete system status.

The STAP Controller also supports various tests to insure proper operation of system hardware. Tests are aimed at the DIU - RCIM and WIM custom designs and the communication paths within the AAP.

The **RCIM** is the RDATA Control Interface Module. This module receives RDATA vectors from the DBF. It also receives control information from the STAP Controller over the VMEbus. The RCIM meters out this data in an implementation specific format to the Arithmetic Processing Units for processing. It also generates timing and control signals. These timing and control signals are derived from the timing signals received from the DBF and the controls received from the STAP Controller. The RCIM is a custom module that contains a VME slave interface.

The RCIM receives timing signals from the DBF, buffers them, and sends them on to yet undefined growth destination. These three signals are the Zero Range Pretrigger (DBFAAPPTRIG), the CPI Trigger (DBFAAPCPI), and the System Clock (DBFAAP8MHZ). The Zero Range Pretrigger signal marks the PRI boundaries. Using this signal, the RCIM can determine the QRD:PRI ratio if this is not specified by the STAP Controller. The RCIM counts 24msecond intervals to determine the number of samples to be processed as part of the QR–Decomposition. Whenever the RCIM detects a change in the PRF, it sends a Reset followed by BIT data for downstream processing. Two PRI's are reserved for this Periodic BIT. This is possible because these two PRIs are fill pulses where no weights are generated by the AAP. The System Clock is used to derive and align timing signals for the RCIM and the WIM.

The RCIM receives RDATA vectors from the DBF interface. One hundred ninety RDATA vectors are always received from the DBF, even though only 186 or less will be used. It stores these vectors in a Dual Port RAM, and strobes this data into the APUs using the signals provided by the RCIM. The RCIM also has a RAM which contains BIT data (RDATA received from STAP Controller), which the RCIM may select instead of the Dual Port RAM data containing DBF RDATA. The RCIM also constructs and sends the Sample Header Word to the downstream processors. The Sample Header Word contains a command to the downstream processors. The commands are: NOP (No operation required - discard data) RESET the RMGSEF algorithm, perform a QRD, Collect Ks and generate adaptive weights, or Break processing.

The RCIM also supports VME interrupts that are triggered from the CPI Trigger. This feature is required to support initial testing of the BSP when the transfer of CPI commands from the STAP Controller must be synchronized to the data stream in the absence of CPI timing from the radar system and Radar Controller.

The **Arithmetic Processing Units** (APUs) are Digital Signal Processing modules (Ixthos' IXZ16) which contain the AAP Computer Software Configuration Item (CSCI). The AAP CSCI performs the RMGSEF algorithm on the data presented to it by the RCIM, and outputs the antenna element weights to the WIM.

The **MMV** APU has been moved into one of the IXZ16 boards and provides an interface between the Link ports and the VMEbus. WTDATA will be supplied to the MMV over a Link port and relayed to the WIM via the VMEbus.

The **WIM** is the Weight Interface Module. It receives the antenna elements from the MMV APU and outputs them to the DBF. The WIM is a custom module that contains a VME slave interface.

The WIM also supports a DBF interface to supply message data in the form of initial weights and Range Sample Addresses. In addition the WIM supports a hard–wired reset to the DBF that can be commanded by the STAP Controller.

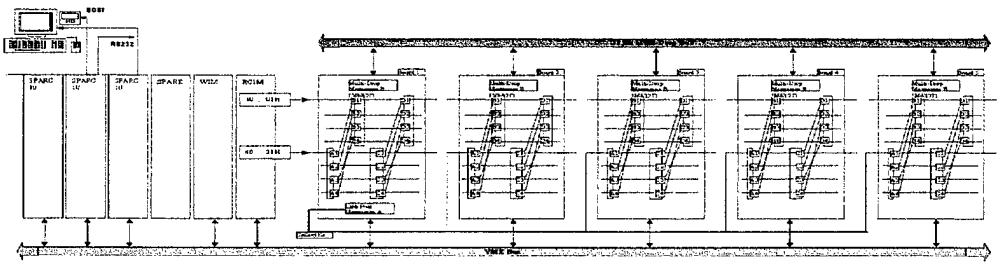


Figure 27 - Ixthos Adaptive Array Processor

7 Digital Beamforming

7.1 Introduction

This section describes the Digital Beamforming that is performed. In addition, various trade studies are included that indicate the possible implementations.

The beamformer produces Sum, Delta and OMNI beams utilizing 18 input data channels. Figure 28 is the functional block diagram of the processing that is performed in the beamformer.

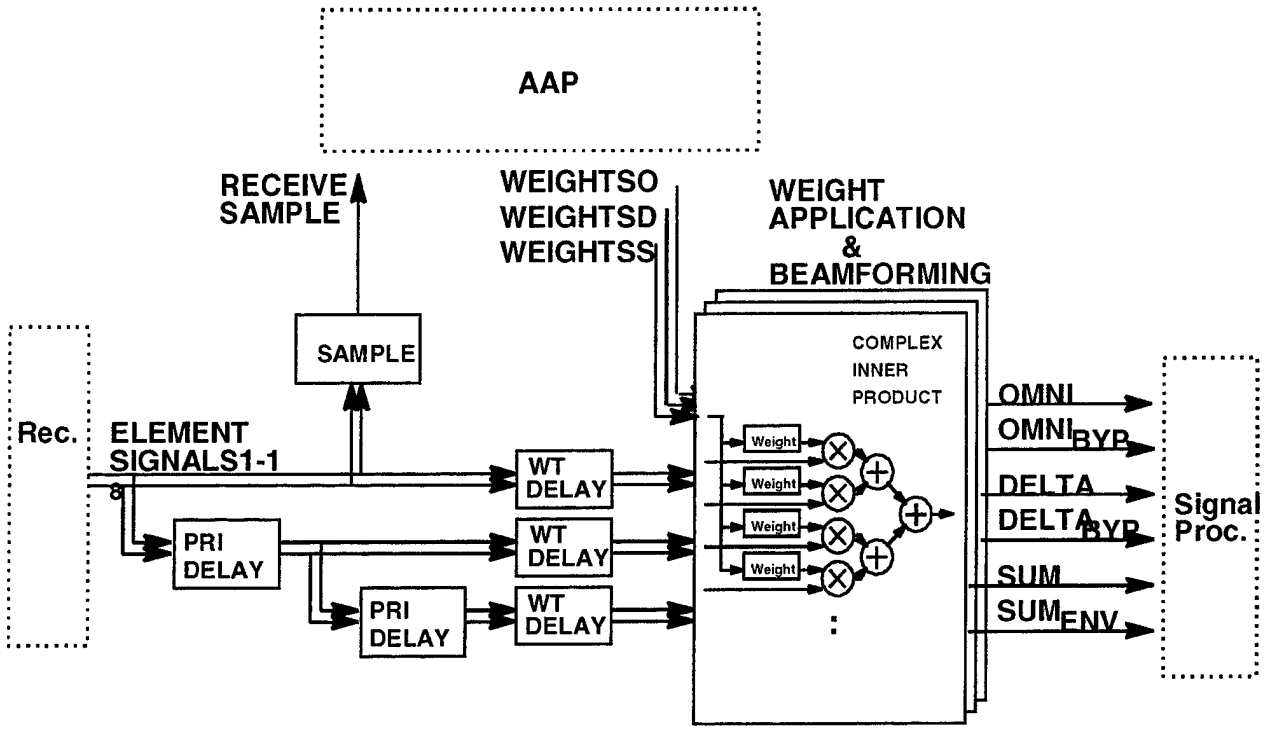


Figure 28 - Beamformer Functional Block Diagram

7.2 DIGITAL BEAMFORMER FUNCTIONAL DESCRIPTION

This section will describe the functions performed in the Digital Beamformer (DBF).

7.2.1 Input Channel Data from the Receiver Subsystem

The DBF accepts inputs from the Receiver Subsystem. Each of the 18 channels send complex I/Q data in 14 bit two's complement format. Each set of 18 channel samples represents one range cell of a PRI.

7.2.2 Sample Processing for Beam Formation

The DBF samples the input channel data of the Receiver. A sample is comprised of 18 complex data values, one from each channel. Samples per PRI are stored until they can be processed and transmitted to the AAP.

7.2.3 Sampling Criteria Definition

The DBF maintains two distinct sampling criteria every PRI to process the channel samples necessary for the adaptive weight calculations in the AAP. Each channel sample contains 18 complex data points from a particular range cell in one PRI. The channel samples must be saved by the DBF for both sampling criteria used for each sector. The DBF transfers 18 channel samples from each sampling period to the AAP every PRI.

7.2.4 Sample Transfer from the DBF to the AAP

Three PRI's are required to form an $\mathbf{r}_s$ data vector. Each $\mathbf{r}_s$ vector consists of 54 complex channel samples, 18 channel samples during the last three PRI's. The number of data vectors sent from the DBF to the AAP is a function of the PRF.

For a 300 Hz PRF, 14,580 (16 bit) words are transferred from the DBF to the AAP. This is an equivalent rate of 4.81Mbytes/s.

7.2.5 Input Data Storage

The input data from the Receiver Subsystem must be delayed in the DBF until the adaptive weights can be applied to it. Memory corresponding to a length of 5 PRI's has been allocated in the DBF for data buffering.

7.2.6 Weights

An adaptive weight vector is a 54x1 complex vector with each complex weight component being a 16 bit two's complement value. Three such vectors are transferred from the AAP to the DBF every PRI, one for each beam. The DBF receives and stores the adaptive weights until they can be applied to the corresponding data vector, **r**. A new weight set could be applied to the complex multipliers every PRI.

7.2.7 Beam Formation

The DBF computes and outputs three primary beams, SUM, DELTA and OMNI. Each beam is a 24 bit two's complement complex vector. The DBF also outputs a single pulse Sum Environmental (SUM_{env}) beam, a Delta Bypass beam and an OMNI Bypass beam, each as a subset of their corresponding primary beam counterparts. The DBF uses the adaptive weights from the AAP along with un-delayed, one PRI delayed and two PRI delayed data vectors to form the beams.

Construction of the three primary beams requires that the DBF perform inner product calculations of either the quiescent weights, **b**, or the adaptive weights, **w**, with the data vectors, **r**. The 54 inner products are summed to form the beam. Scaling is performed to convert the beam output to a 24 bit fixed point format.

7.2.8 Beamforming Algorithm

The DBF utilizes the following Beamforming algorithms to process the 18 input channel samples into the three primary beams (Sum, Delta, Omni).

$$S_n = \sum_{x=1}^{x=18} W_{\Sigma}(x)\, R_n(x) + W_{\Sigma}(x+18)\, R_{n-T}(x) + W_{\Sigma}(x+36)\, R_{n-2T}(x)$$

$$D_n = \sum_{x=1}^{x=18} W_d(x)\, R_n(x) + W_d(x+18)\, R_{n-T}(x) + W_d(x+36)\, R_{n-2T}(x)$$

$$O_n = \sum_{x=1}^{x=18} W_o(x)\, R_n(x) + W_o(x+18)\, R_{n-T}(x) + W_o(x+36)\, R_{n-2T}(x)$$

where

S_n equals sample n of the Complex Output Sum Beam,
D_n equals sample n of the Delta Beam,
O_n equals sample n of the Omni Beam,
x equals the input channel number,
W_{Σ} . W_d . W_o represent the conjugate of the weight vector over 3 PRIs (54 complex weights per beam) calculated by the AAP.
Rn(x), Rn-T(x), Rn-2T(x) represents the receive channel samples from the 3 PRIs with n representing the range bin number within the PRI and T is the number of r vectors (range bins) per PRI.

S_n, D_n and O_n are formed by summing the weighted input signals over 3 phase centers (3 PRI).

7.2.9 Sum Environmental Beam

The fourth beam, Sum_{env} is a by-product of the Sum Channel construction. Sum_{env} is formed from 18 of the 54 **w** and **r** values. The set of **wr** values are either the sum of the un-delayed inner products, or the sum of the one PRI delayed inner products or the sum of the two PRI delayed inner products of the Sum Channel. The selection is based on a control word from the STAP Controller. Scaling is performed to convert the beam output to a 24 bit fixed point format.

7.2.10 Delta Bypass Beam

The fifth beam, $Delta_{byp}$ is a by-product of the Delta Channel construction. $Delta_{byp}$ is formed from 18 of the 54 **w** and **r** values. The set of **wr** values are either the sum of the undelayed inner products, or the sum of the one PRI delayed inner products or the sum of the two PRI delayed inner products of the Delta Channel. The selection is based on a control word from the STAP Controller. Scaling is performed to convert the beam output to a 24 bit fixed point format.

7.2.11 Omni Bypass Beam

The sixth beam, $Omni_{byp}$ is a by-product of the Omni Channel construction. $Omni_{byp}$ is formed from 18 of the 54 **w** and **r** values. The set of **wr** values are either the sum of the undelayed inner products, or the sum of the one PRI delayed inner products or the sum of the two PRI delayed inner products of the Omni Channel. The selection is based on a control word from the STAP Controller. Scaling is performed to convert the beam output to a 24 bit fixed point format.

7.3 DBF IMPLEMENTATION TRADE STUDIES

Two implementation solutions were studied for performing the beamforming. The first was based on utilizing the HPSC modules assuming 32 processors per module and the second was based on an FPGA solution.

7.3.1 DBF Requirements Summary

The following are the driving requirements in the implementation of the DBF algorithm.

- 18 channels
- 3-Pulse Temporal Delay
- Sum, Delta, & Omni Beams
- 9 x 18-weight Complex Inner Products (162-weights)
- 8MHz Sample Rate

- 16-bit Fixed Point Multiplies
- Accommodate Summation Bit Growth

7.3.2 SHARC DSP Characteristics

The following are the characteristics that are used to estimate the number of DSP processors required to implement the DBF processing using the HPSC modules.

- CLOCK SPEED = 40Mhz
- CYCLE TIME = 25 NANOSECONDS
- 3 FLOPS/CYCLE = 120MFLOPS Peak
  *** Supported by FFT Instructions ***
- 2 FLOPS/CYCLE = 80MFLOPS Peak
  *** Supported by Inner Products ***

7.3.3 HSPC DBF Sizing Estimate

Table 7 contains estimates of the number of operations required to implement the DBF.

Table 7 - DBF Processing Load

FUNCTION	EQUATION	MOPS	EFFICIENCY	# OF PROCESSORS
Formatting	8MHz*(Smpls)*[(18*2)]*(Assmb)*+[(18*2)]*(Mpys)*	576	576 / (80Mflops*25% EFFICIENCY)	29
Complex Inner Products	8MHz*(Smpls)*[[(54*6)]*(Mpys)*+[(54-1)*2]*(Adds)*]*3(Beams)*	10320	10320 / (80Mflops*57% EFFICIENCY)	226
Control Overhead	*255(# of processors)* + 30%			**TOTAL = 331**

An HPSC node consists of four SHARC processors resulting (331 Processors / 4) = 83 Nodes

7.3.4 DBF - Design Trade Study

From this information it will be possible to evaluate several possible implementations. The following are the primary driving factors in the establishment of the DBF architecture utilizing the HPSC modules.

- Distributing 576MB/s to 331 processors while minimizing the processing latency.
- MTI Function dictates recursion.
- Partitioning data by Beams – 576MB/s is tripled to 1.728GB/s.
- Partition Data by Range (preferred approach).

7.3.5 DBF - HPSC Implementation

Figure 29 represents the system implementation showing the driving factors.

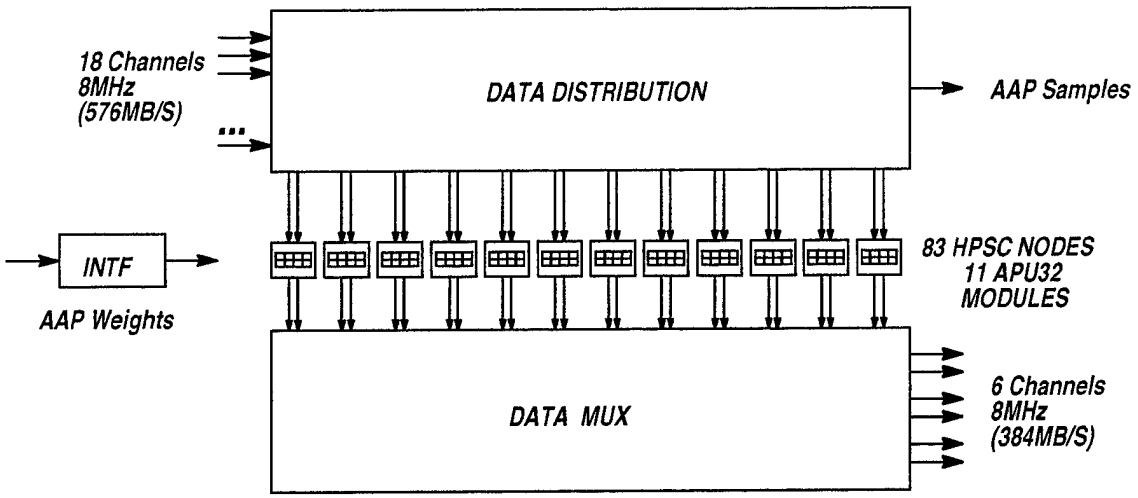


Figure 29 - DBF HPSC System

As can be seen from Figure 29 above, the distribution of the data to the HPSC modules is a driving factor in the architecture. Figure 30 shows an implementation utilizing the HPSC modules and a Myrinet™ Network for the distribution of the data.

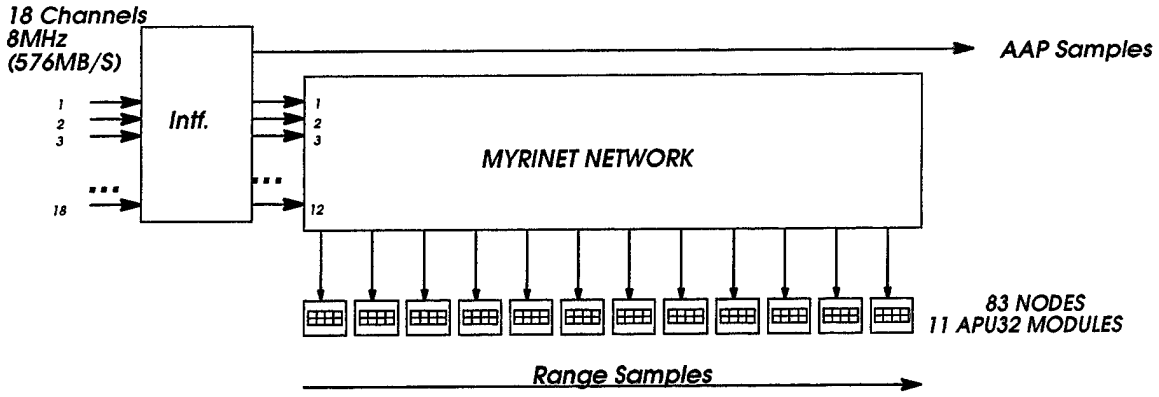


Figure 30 - HPSC DBF Architecture

With this architecture, the following are the requirements for the data transfer.

- 83 Range Data Sets are required to be transferred every PRI = 1 Range Data Set every 31us.

The following are the measured HPSC Myrinet™ transfer and latency times.

- Processor to Processor - Myrinet™ TRANSFERs

Latency = 120us. Throughput = 25 MB/s.

- Off board - Myrinet™ TRANSFERs

Latency = 60us. Throughput = 50 MB/s.

If the data is gathered into large blocks, the Myrinet™ overhead can be reduced. For example gathering the data as follows allows the transfer requirements to be met.

8x83 Range Data Sets are required to be transferred every 8xPRI or 21ms = *8 Range Data Set every 253us*

8x21052 Range Bins*4 Bytes(I&Q) / 83 Data Sets = 8116 Bytes/Data Set.

8116 Byte Myrinet™ message requires = 222us by each node.

In addition to the data, 83 weight sets need to be transferred to the DBF from the AAP every 31us. The time for transferring the data for a single node is as follows:

162 Weights x 8 Bytes(I&Q) = 1296 Bytes/Weight Set

1296 Byte Myrinet™ messages requires = 86us by each node.

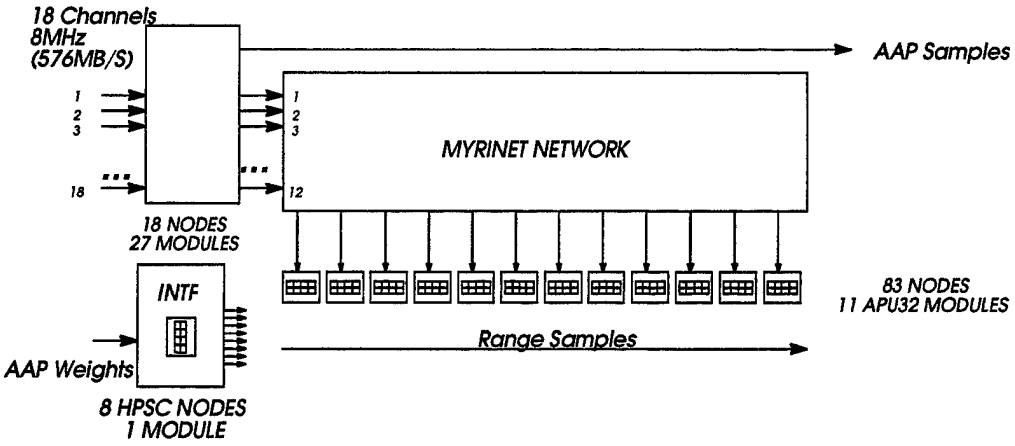
As can be seen a single node can not transfer the weights in the allotted time.

By again grouping the data into 8 sets per transfer and re-computing the transfer time leads to the requirement of being able to transfer the weights in 8x31us or 253us. From this the transfer times are re-computed as follows:

162x8 Weights x 8 Bytes(I&Q) = 10368 Bytes/Weight Set

10368 Byte Myrinet™ messages requires = 267us by each node.

As can be seen, even with regrouping the data into a larger block it is still not possible to transfer the data in the allotted time. One solution to transfer the data would be to utilize 8 HPSC nodes and distribute the data in a serial to parallel approach. Figure 31 shows this method.



Weights required to be distributed among nodes who in turn perform *redistribution (similar to "chain letter")*

Figure 31 - Parallel Weight DistributionNew MCM Technology

New 8 NODE (32 processor) MCM module would utilize 3V technology to reduce power and would also incorporate a 16 way switch. In addition a new Myrinet™ Interconnect Module that incorporates a 16 way switch would be required to increase the number of buffered I/O's. Finally a new backplane design would required to support this new technology. Utilizing this new technology, the size of the new DBF can be estimated.

Table 8 - DBF Module Count using MCM's

Function	Module Type	Quantity
HOST	COTS	1
Data Distribution	Custom I/O	19
	APU_{32}	4
	$MTEM_{16}$	4
Beamforming	APU_{32}	11
	$MTEM_{16}$	4
		TOTAL = 43

From this a size/weight/power/cost comparison can be drawn and this is depicted in Figure 33.

A large investment would be required to develop this level of packaging integration. Advances in processor technology would accomplish the same level of overall processing density.

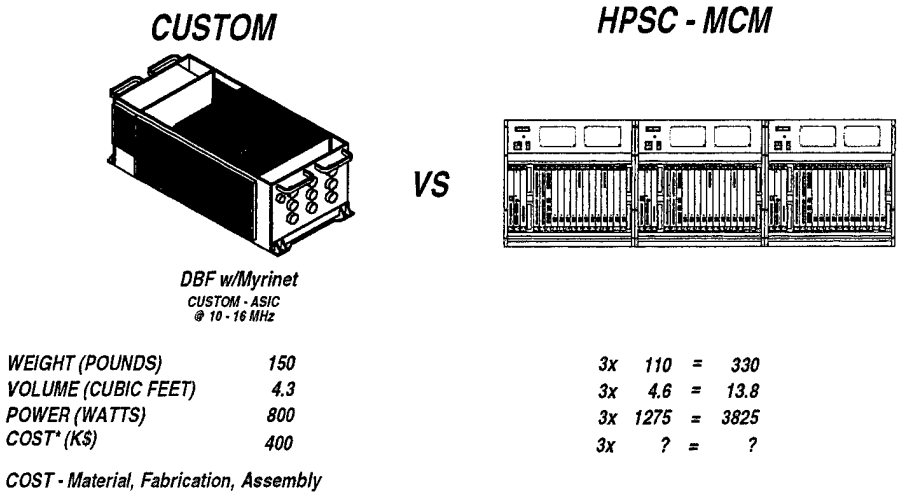


Figure 32 - CURRENT versus HPSC MCM comparison

7.3.6 DBF SOLUTIONS

The following sections describe various approaches that have been studied for the implementation of the beamformer.

7.3.6.1 CURRENT DIGITAL BEAMFORMER DESIGN

Figure 34 represents the current beamformer design

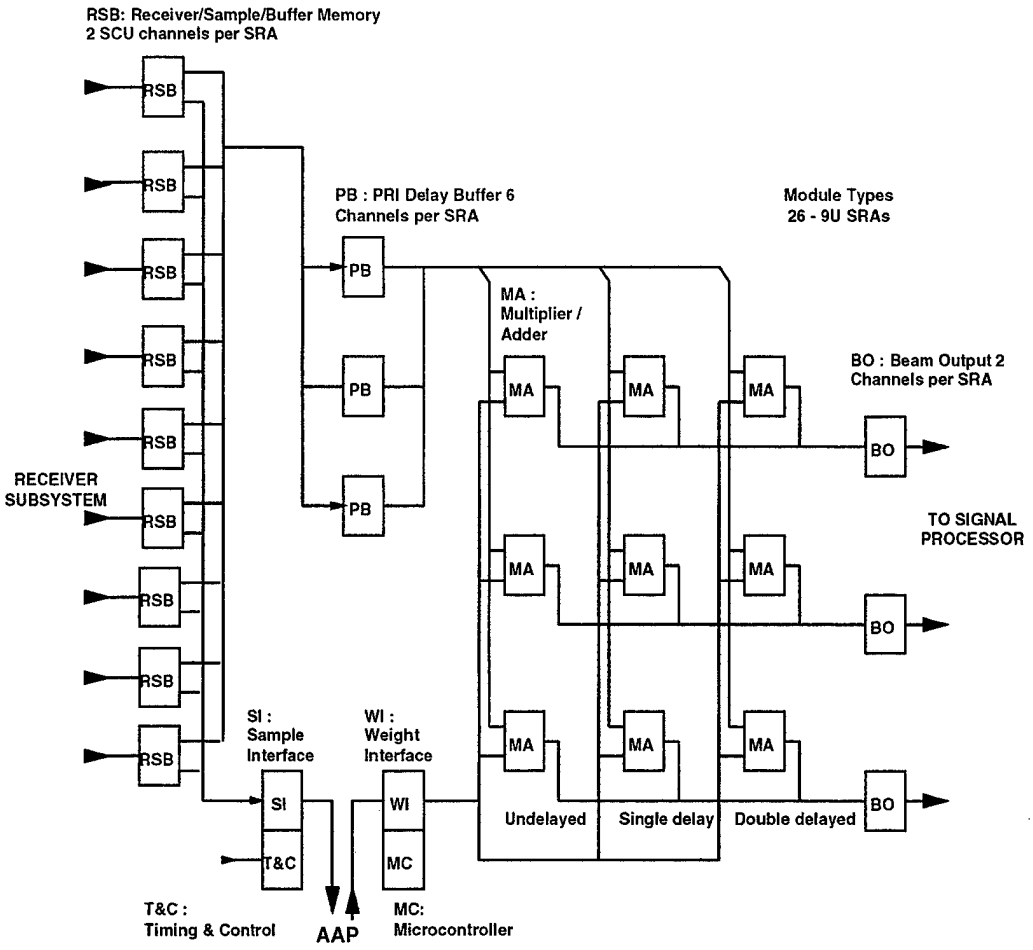


Figure 33 - Current Custom DBF Implementation

The current DBF employs 26, 9Ux220 boards. The 9 receiver Sample Buffer Memory (RSB) boards receive data from the Receiver, 2 channels per board. The Receiver interface is RS422 using parallel, differential, twisted pairs. The received data is sampled, collected by the Sample Interface (SI), and sent to the AAP for weight generation. The weights computed by the AAP are received by the Weight Interface (WI), and distributed to the Multiply Add (MA) boards. The sample and weight interfaces are the same as the Receiver interface: RS422 using parallel, differential twisted pairs. After the transient delay on the RSB, the receiver data is applied to the PRI Delay Buffers, which hold 6 channels per board. The PB boards generate the un-delayed, one delayed and two delayed data streams that are input to the Multiply Add boards. Each MA board contains 18 Butterfly chips that perform the complex multiply, and the adder tree that generates the part beam sums. The Beam Out modules sum the 3 part beams to form the output beams which are sent to the Signal Processor. The Signal Processor interface is the same as the others: RS422 using parallel, differential twisted pairs.

7.3.6.2 DBF II CONCEPT 1

DBF II concept 1 employs 4 board types for a total of 10 boards. The Receive Sample Buffer memory and Delay (RSBD) board receives data from 4 receiver channels, and generates 4 sets of Undelayed, One delayed and Two delayed data streams. The Multiple Add and Beam Output (MABO) boards form the complex inner product and additions to generate 2 output beams per board. The Sample Interface Timing and Control (SITC) and Weight Interface and Micro Controller (WIUC) boards perform the same functions as they do in the current design. The Receiver and Signal Processor interfaces are Hewlett Packard high speed serial (HDMP), while the interface to and from the AAP is Myrinet™. Figure 35 depicts this concept.

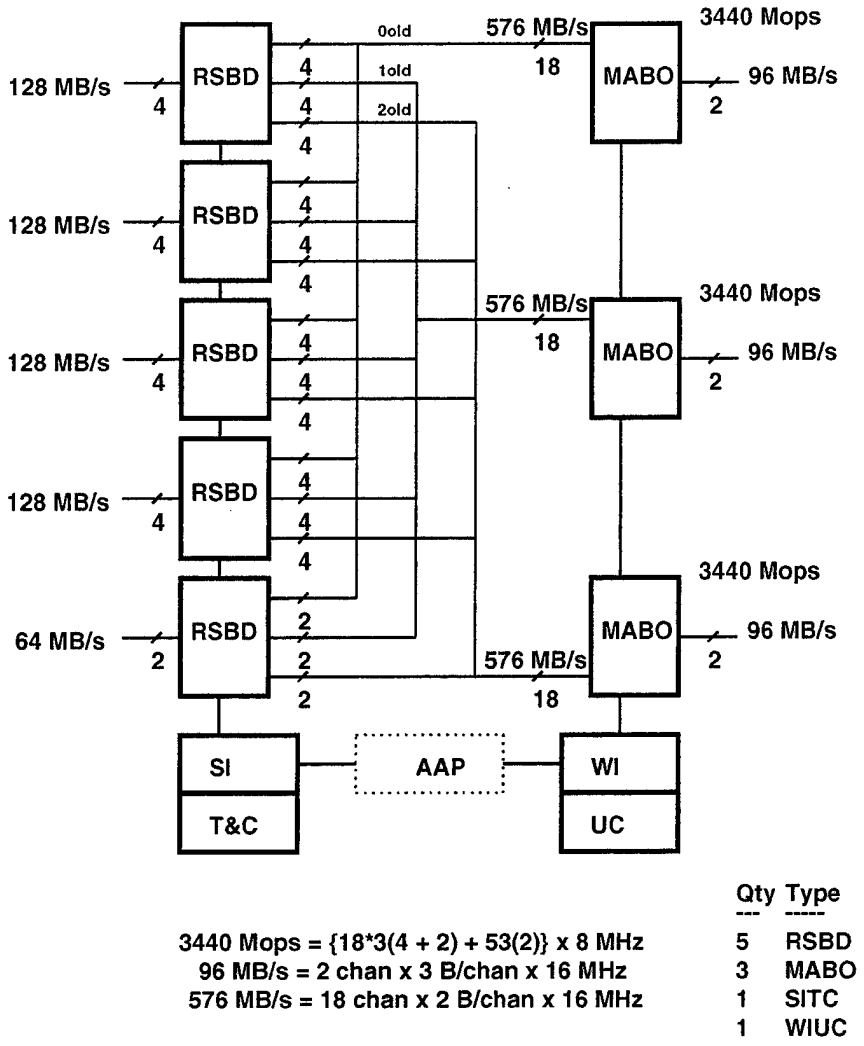
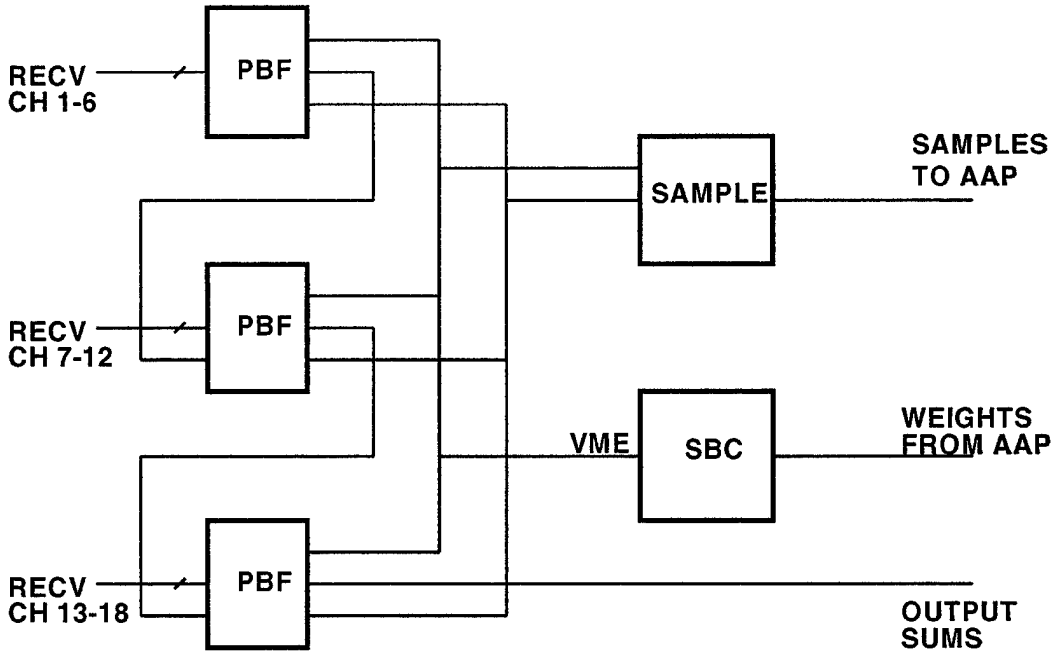


Figure 34 - DBF Concept 1

7.3.6.3 DBF II CONCEPT 2

The DBF II concept 2 will consist of three board types with a total of 5 to 8 boards depending on the final requirements. The first board type will be a custom designed Partial Beamformer. This board will use the latest technolgy in FPGAs, and memory to perform the entire beamforming process for some subset of channels (2-6) on one board. The second board type will be a custom designed sample interface which will collect the sample data and send it to the AAP. The third board will be a COTS single board computer to simplify software development and simplify system control interfaces.

DBF II will use high speed serial (HDMP) interfaces to cut the number of wires between the DBF and the receiver from over a thousand to less than 10. It will also use high speed serial (HDMP) interfaces to pass data between boards. This will avoid the large number of backplane interconnects, over two thousand on the existing DBF. DBF II will also utilize the VME bus to eliminate the custom backplane. Figure 36 represents this concept.



VME based with data transfers via high speed serial interfaces

SBC - Motorola 167 single board computer or similar COTS board
PBF - Partial beamformer - beamform up to 6 channels at 5 MHZ
SAMP - Sample board - will collect and send samples to AAP

Figure 35 - DBF Concept II

8 Receiver Switching Network Study

A portion of the Receiver Switching Network study involved economizing interconnect in the overall string from Receiver through Digital Beamformer to Signal Processor. This section discusses an upgrade within the AEW Signal Processor to not only get Beamform data into the Signal Processor chassis more efficiently, but to also get selected samples of that data onto the Signal Processor's Myrinet™ network more promptly and straightforwardly.

The present Digital Beamformer (DBF) connects to the Signal Processor using standard, circular 128-position plugs connecting to 128-position, bulkhead-mounted, circular receptacles. Within the Signal Processor, a cable harness assembly is used to route the incoming DBF data signals to Custom Input Channel cards.

A single slot 6Ux160 VME board, the Myrinet™ Custom Interface (MYCI) is under development which receives two channels of DBF data by way of front panel 128-pin (4-row by 32) connectors. The MYCI provides connectivity to a Myrinet™ network over two Myrinet™ "links" for two custom channels of data (where each custom channel has its own dedicated link) in the same single VME 6U slot.

The MYCI is designed to interface at its front panel with two custom channels of 64-bit data (32-bit I and 32-bit Q values), updated at a 5 MHz rate. This equals 320 Mbps or 40 MB/s for each custom channel. Each custom channel has its own, dedicated byte-serial Myrinet™ Link, which is capable of 320 MBytes/sec or 160 MB/s full-duplex (i.e., 160 MB/s over each

of a Myrinet™ link's two "channels": Myrinet™ Channel In*, Myrinet™ Channel Out*). The two Myrinet™ links are routed to a single System Area Network (SAN) connection, which, physically, is the VME64 Extensions (VME64x) P0 connector.

The custom channel interface logic and its control of the custom channel side of the swing-buffered memory is implemented in a FPGA on a daughter card so that different interface options and media can be accommodated. For example, one custom interface could be parallel, differential receive, while the other could be parallel, differential transmit. Similarly, two custom channels of serial receive could be connected to the MYCI.

Between each custom channel and its Myrinet™ link there is a swing-buffered and dual-ported memory bank. The depth of the buffer memory allows a queuing of one block of samples (up to 16,384) within a Pulse Repetition Interval (PRI). Reads of the memory are therefore accessing the prior PRI's collected samples.

The two Myrinet™ link interfaces each have a custom VLSI chip from Myricom, Inc. to provide the physical interconnect with Myrinet™. The MYCI has an FPGA between the Myricom custom VLSI chip and the swing-buffered memory, in order to handle memory transfers and any chip-specific interaction.

The MYCI also has a slave VME backplane interface for use in simple diagnostics, parameter loading, status access, custom channel enabling or disabling and Myrinet™ link enabling or disabling.

* NOTE: To avoid confusion, 'channel' is meant to be synonymous with 'custom channel' in this document. The two channels that comprise a Myrinet™ 'link' will be called 'Myrinet™ Channel Out' and 'Myrinet™ Channel In' whenever it is necessary to mention them. The MYCI's output to Myrinet™ is called 'Myrinet™ Channel Out' and the MYCI's input from Myrinet™ is called 'Myrinet™ Channel In'.

9 Evaluation of ISI RTExpress Tool

RTExpress is a tool that translates/compiles MATLAB code to real time parallel code on embedded and high performance computers. This tool utilizes Message Passing Interface (MPI) the de facto industry standard for communications between nodes. It also provides a convenient suite of tool's which simplifies the generation of real time code.

The target balancing tool provides simple methods of breaking code up into computational units or groups, assigning nodes to these groups and specifying the type of parallelism (data parallel, round robin, etc.). The user need only specify parallelism at a top level without reference to the extreme level of detail normally associated with the development of parallel applications.

The compilation process links in the appropriate machine specific vector libraries, MPI, SCALAPACK and RTExpress parallel function libraries, resulting in vectorized code for the specific target architecture.

Real Time Performance Monitoring provides detailed monitoring of processor loading and communications which facilitates balancing node assignments. A very convenient feature is the ability to change node assignments without recompilation. By iterating the process of node assignment and performance monitoring processor loading can be optimized in a convenient, straightforward and expeditious fashion. Although the use of dialog boxes for node assignment in the target balancing tool is straightforward, a graphical tool, Mapit, is under development which will produce a graphical display of the target architecture and allow graphical assignment of nodes to groups, further facilitating the development process.

In addition to its primary application of producing real time mission code an immediate application of this tool is to take MATLAB simulation and analysis tools and transfer them to the target architecture resulting in a sophisticated data analysis capability which should further enhance the development process. Further evaluation of this tool is to be performed in the future to further explore these possibilities.

10 Conclusions

Initial mapping and partitioning studies determine that an HPSC hardware configuration employing 716 SHARC PEs can support simultaneous Pre and Post Doppler STAP processing architectures; the selection is accomplished through on-the-fly software re-programmability of data routing tables. It may be noted that this solution utilizes two identical chassis and only two board types. These algorithms represent sustained processing throughputs of approximately 32 Gflops.

Further detailed algorithm mappings investigate the parametric latencies and efficiencies of three different mappings of the RMGSEF QRD portion of the STAP processing. This processing is relatively complicated and fine-grained in terms of data flow and interdependencies between individual processing elements. For a particular data point in terms of Myrinet™ and SHARC latencies and throughputs, simulations revealed that the sample recursive Myrinet™ only solution offers better than a

2:1 computational latency advantage over the block recursive Myrinet™ only mapping. The addition of SHARC links to the mapping in order to create a much finer grained processing solution further reduces the computational latency due to the small message sizes and low communication latency, however, at the cost of additional hardware support for the SHARC links. The study also revealed that as the Myrinet™ overheads are parametrically reduced and communication delays are improved, the three mappings offered approximately the same performance.

The CSIM tool proved valuable as a system architecture development tool. Using the simulation outputs, a possible solution space of mappings is defined, in terms of network parameters, radar PRF, mapping approach (block versus sample recursive) and mapping granularity. As well, these results can further be used to tune the algorithm mapping as the Myrinet™ performance continues to improve.

The demonstration proved that the HPSC architecture is highly flexible supporting high processing loads with minimal latency.

The follow on Architecture study revealed that the CSPI SHARC based module (2316) does not incorporate the SHARC link ports, as did the HPSC. Thus CSPI does not offer a direct replacement for the AEW Adaptive Array Processor. The SHARC based Ixthos modules offers in addition to the link ports a multi-drop bus that is limited in terms of its scalability. While the SHARC has remained at 40MHz the processing world has seen the speed of the PowerPC grow to 300MHz with architecture enhancements translating into competitive processing alternatives to historical Digital Signal Processor application.

Digital Beamforming however is an area where fixed-point arithmetic makes processor based solutions very inefficient. The processing density of Field Programmable Gate Arrays offers significant advantages in size, weight and power. VHDL based designs provides for portable solutions.

CSPI offers a set of building blocks to support interfacing to sensor data. One approach is to bridge from a Front panel Data Port (FPDP) to Myrinet™ via a PCI bus. This solution presently occupies two card slots with single card slot solutions on the horizon. Leveraging RCTE, a more efficient solution with reduced latency and greater flexibility can be realized.

The latency study is as yet inconclusive and will be continued under IR&D. Benchmarks indicate that for this algorithm the SHARC operating at 32.5MHz is comparable to the 200MHz PPC-603. The improvements in the next generation PPCs indicates that a fewer number of boards each containing four PPC-750s are required to support the data throughput. The realized latency is expected to be comparable to the HPSC solution.

Continuation of this study will concentrate on the CSPI/Myrinet PPC technologies.

11 Acknowledgement

The HPSC technology is under joint development with Sanders, a Lockheed Martin company, under sponsorship from DARPA ITO in conjunction with USAF Rome Laboratory.

12 References

[1] "Embedded HPSCS", Sanders, http://www.sanders.com/hpc/HPSCS/HPSCS.html

[2] "Myrinet™ - A gigabit-per-second local area network", IEEE Micro, February 1995; http://www.myri.com/research/index.html

[3] Ling, F., Manolakis, D., and Prokis, J. G. (1986) "A recursive modified Gram-Schmidt algorithm for least-squares estimation." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34, 4 (Aug. 1986), 829-836.

[4] "ADSP-2106x SHARC User's Manual", E1995 Analog Devices, Inc.

[5] "IXLibs-21k User's Manual", E1995 Ixthos, Inc.

[6] "RASSP", Lockheed Martin Advanced Technology Laboratories, http://www.atl.external..lmco.com/projects/rassp/rassp.html

13 Biographies

Ronald E. Hamlet

Mr. Hamlet received a BSEE degree from Michigan Technological University in 1979, and an MSEE degree from Syracuse University in 1982.

Mr. Hamlet began working for G.E. at the Aerospace Electronic Systems Department in Utica, NY in 1979. His work has concentrated on advanced signal processing techniques using custom very high speed integrated circuits as well as commercial-off-the-shelf technologies. He has applied these solutions in the areas of Infrared sensors and Radar.

Recently, Mr. Hamlet has worked on Space Time Adaptive Processing and its application to AEW radar. He is the principal investigator for the HPSC program at Syracuse Lockheed Martin.

DISTRIBUTION LIST

addresses	number of copies
AFRL/IFTC RALPH KOHLER 26 ELECTRONIC PARKWAY ROME NY 13441-4514	2
LOCKHEED MARTIN OCEAN RADAR AND SENSOR SYSTEMS PO BOX 4840, ELECTRONIC PARK SYRACUSE NY 13221-4514	1
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P.STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/HESC-TDC 2698 G STREET, BLDG 190 WRIGHT-PATTERSON AFB OH 45433-7604	1

ATTN: SMDC IM PL
US ARMY SPACE & MISSILE DEF CMD
P.O. BOX 1500
HUNTSVILLE AL 35807-3801 1

COMMANDER, CODE 4TL000D
TECHNICAL LIBRARY, NAWC-WD
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100 1

CDR, US ARMY AVIATION & MISSILE CMD
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSAM-RD-OB-R, (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5000 2

REPORT LIBRARY
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545 1

ATTN: D'BORAH HART
AVIATION BRANCH SVC 122.10
FOB10A, RM 931
800 INDEPENDENCE AVE, SW
WASHINGTON DC 20591 1

AFIWC/MSY
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016 1

ATTN: KAROLA M. YOURISON
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE
PITTSBURGH PA 15213 1

USAF/AIR FORCE RESEARCH LABORATORY
AFRL/VSOSA(LIBRARY-BLDG 1103)
5 WRIGHT DRIVE
HANSCOM AFB MA 01731-3004 1

ATTN: EILEEN LADUKE/D460
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730 1

DUSD(P)/DTSA/DUTD 1
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

MISSION OF AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.